

DECEMBRE 1987

LE JOURNAL QUI N'A PAS BESOIN D'OXYGENE POUR RESTER JEUNE

TURBO FORTH

it's

MAD

EDITORIAL

Ouf, je lève un peu le nez de TURBO-Forth pour vous finir ce numéro. Déjà les premières réactions des adhérents ayant reçu la version d'évaluation sont très favorables. C'est qu'il n'est pas évident du tout d'imposer un nouveau standard, car c'est bien là notre but final.

Si comme je l'espère, une revue grand public diffuse la version d'évaluation sous film plastique et joint à la revue, alors TURBO-Forth deviendra la version du langage FORTH la plus répandue en France.

Car jusqu'à présent, nous nous heurtions à un mur: comment intéresser les programmeurs à un langage dont personne ne parle, comment diffuser des informations concernant un langage que très peu de gens pratiquent. C'est un peu l'histoire de la poule et de l'oeuf: qui a pondu qui.

Pour briser ce cercle vicieux, une seule solution, la diffusion massive à un prix ridicule. En créant le marché, on génère l'intérêt, le développement de programmes, les utilitaires et les applications, la documentation. Mais cette politique nous fait renoncer en tant qu'auteurs de cette version (ZUPAN et PETREMANN) aux hypothétiques droits qu'une commercialisation normale nous aurait apporté. Adieu la piscine au bord de la villa à Sun Valley.

Notre seul bénéfice et satisfaction sera l'intérêt et le succès que remportera cette version sur toutes les autres. Alors essayez TURBO-Forth et dites-nous ce que vous en pensez, que nous ne soyons pas les seuls à louer ses qualités.

Et notre nouvelle formule:

"MAY BE HAVE THE TURBO-Forth WITH YOU"
(Luc SKYWALKER dans LA GUERRE DES ETOILES - G.LUCAS)
(adaptation personnalisée: revue JEDI)

SOMMAIRE

Thème de couverture emprunté au PETIT JOURNAL (encart TELERAMA n° 1982, janvier 1986)

Nos coordonnées: ASSOCIATION JEDI, 17, rue de la Lancette 75012 PARIS
tel président: (1) 43.40.96.53
tel secrétaire: (1) 46.56.33.67

TURBO-Forth ENCORE!!!

par M. PETREMANN

Il y a une publicité passant dernièrement à la télévision où un homme des cavernes essayait de satisfaire sa gourmandise par des procédés risqués jusqu'à ce qu'il rencontre une créature de rêve et une délicieuse friandise. Anachronisme aidant, celle-ci était en voiture (rouge je crois). Le dernier plan montre cet homme visiblement insatisfait se tapant sur le crâne en disant: ENCORE, ENCORE... C'est précisément la réaction que vous aurez après avoir lu ce chapitre vous présentant TURBO-Forth.

Vous aimez programmer en FORTH, mais les blocs vous coincent. Le standard F83 de Laxen et Perry avait un besoin urgent d'être dépoussiéré et, surtout, de pouvoir digérer les fichiers ASCII. C'est pourquoi, en partant de la fonction INCLUDE diffusée dans un précédent numéro, l'idée nous est tout naturellement venue de créer, vous savez quoi?... ben TURBO-Forth voyons!.

Pourquoi le préfixe TURBO? Et pourquoi-pas! Qu'est-ce qui a fait la célébrité de TURBO-Pascal? TURBO-Pascal dispose certes d'un compilateur très performant, mais surtout, il brise le cycle infernal ECLR (Edit Compile Link and Run). F83 aussi brisait ce cycle, mais ses possibilités d'édition de fichiers un peu trop exotiques rebutaient plus d'un amateur.

Enfin TURBO-Forth est arrivé. Ce nouveau FORTH n'est rien de moins que la version Laxen et Perry (ne jetez pas votre manuel), mais toutes les fonctions de gestion et d'édition de bloc en ont été supprimées. F83 fait 26K de taille mémoire, TURBO-Forth ne fait plus que 24k. Pour vous allécher un peu, voici les fonctions essentielles et nouvelles qui équipant TURBO-Forth:

INCLUDE <fichier[.ext]>
Cette fonction est réentrante et peut traiter jusqu'à cinq fichiers (si votre CONFIG.SYS contient la ligne FILES = n, où n est supérieur à 8, ce nombre peut être augmenté). Ainsi, un fichier peut en appeler un autre qui en appelle un autre, le retour en fin de compilation de fichier renvoie à la poursuite de la compilation du contenu du précédent fichier. En cas d'erreur de compilation, un message indique le n° de ligne et le nom du fichier où s'est produite l'erreur. Ces paramètres sont sauvegardés et par appui sur la touche F2, vous pouvez passer en éditeur ASCII plein écran.

EDIT
Cette fonction est appelée automatiquement par appui sur F2. Si une erreur s'est produite en cours de compilation, l'éditeur charge le fichier à corriger et place le curseur à la ligne et colonne où s'est produite l'erreur. Exemple, soit le fichier TEST.FTH contenant le très court programme:

```
: BOUCLE ( ---)
 100 0
     DOI . LOOP ;
```

Si vous compilez le contenu par INCLUDE TEST (l'extension .FTH est rajoutée automatiquement; pour une autre extension, la préciser), le message d'erreur affichera:

```
DOI . LOOP ; DOI inconnu

DOI . LOOP ;
   ^
```

Ensuite, l'appui sur F2 déclenche la procédure d'édition et positionne le curseur sur la partie erronée. Luxueux non? Et si vous voulez utiliser votre traitement de texte favori, mettez son nom dans la variable chaine EDIT$:

```
" WS.COM" EDIT$ $!
```

L'appui sur F2 lance WORDSTAR,

```
" WP.EXE" EDIT$ $!
```

L'appui sur F2 lance WORDPERFECT

CHDIR chemin
Cette fonction permet de sélectionner un autre sous-répertoire que le sous-répertoire courant. Ainsi, si vous démarrez TURBO depuis le répertoire principal, vous pouvez sélectionner le répertoire contenant dBASE (pour exemple):

```
CHDIR \DB3P
DIR
  ... affiche directory du répertoire DB3P ...
```

DIR [d:][path][masq.ext]
Affiche le contenu de la directory courante, y compris la taille de chaque fichier. Le seul mot DIR agit comme DIR *.* Toutes les options et masques sont possibles comme sous MSDOS (DIR *.COM, DIR T???.FTH, etc...).

PROGRAM <prog.COM>
Lance l'exécution d'un programme extérieur à FORTH. Cette fonction peut être utilisée dans une définition compilée de la manière suivante:

```
: BOUCLE 10 0 DO
 CR I . I 5 =
 IF " PROGRAM WS.COM " $EXECUTE  THEN
 LOOP ;
```

Ce qui agit de la manière suivante:

```
0
1
2
3
4
5
   lancement de WORDSTAR
   ... sortie de WORDSTAR ...
6
7
8
9
```

Les paramètres sont passés au programme appelé par PASS, exemple:

```
" ESSAI.FTH " PASS PROGRAM WS.COM
```

et qui équivaut depuis MSDOS à la séquence:

```
WS ESSAI.FTH
```

Le mot PROGRAM résout en outre une question fréquement posée et qui concerne un véritable FORTH travaillant dans tous les segments. Avec PROGRAM, votre module principal peut exploiter une quantité de sous-modules sous forme compilée. En fonction de la taille mémoire disponible (de 256k au moins à 640k) vous pouvez héberger de un à quatre programmes au moins en plus du programme principal. Cependant, le lancement de certains programmes exploitant "sauvagement" des zones de mémoire situées en dehors de leur segment de chargement peuvent planter votre programme principal.

Mais par contre, PROGRAM n'interdit pas, bien au contraire, d'exploiter des sous-modules compilés par d'autres langages (TURBO-Pascal, TURBO-Prolog, dBRUN, C, BASIC compilé, etc...). L'utilisation d'un sous-module PASCAL ou dBASE permet de résoudre certains problèmes par l'emploi des nombres en virgule flottante exploités par ces langages. Inversement, si vous démarrez par exemple sous dBASE, vous pouvez lancer un programme FORTH. Exemple:

- un programme dBASE exploite un fichier et génère un

état des données sous forme condensé dans un fichier ASCII (option ALTERNATE TO fichier et ALTERNATE ON);

- dBASE lance un programme TURBO-Forth compilé gérant par exemple des fonctions graphiques; ce programme ouvre le fichier contenant l'état des données généré par dBASE;

- les données sont traitées par FORTH (tracé d'histogramme, camembert, etc...), réalise une impression graphique et redonne la main à dBASE.

- dBASE termine son travail et ferme ses fichiers.

Vous croyez que c'est difficile? Eh bien, si vous êtes malin, pensez à programmer dBASE pour qu'il génère des lignes de texte directement interprétables par FORTH. Pour exemple, soit un véhicule dont on désire connaitre la courbe de consommation annuelle de carburant et la visualiser par histogramme. Votre programme dBASE ouvre le fichier des consommations, demande le n° minéralogique du véhicule et l'année à éditer:

NOMIN Numéro minéralogique, caractères
CARBUR Quantité de carburant en litres, numérique, 2 décimales

Une boucle de 1 à 12, sous dBASE va concaténer une ligne de texte ASCII:

```
SET ALTERNATE TO HISTO.FTH
...option SET MASK...
STORE ' " '+NOMIN+'" NOMIN$ $! ' TO LIGNE
STORE 1 TO MOIS
DO WHILE MOIS<12
     ..sélection consommation du mois pour véhicule courant...
   STORE LIGNE+STR(CARBUR,6,2)+" " TO LIGNE
ENDDO
SET ALTERNATE ON
? LIGNE
STORE " " TO LIGNE
SET ALTERNATE OFF
RUN TURBO HISTO
```

à partir de là, TURBO-Forth démarre avec comme paramètre l'exécution du mot HISTO. Ce mot ouvre le fichier HISTO.FTH et y trouve la ligne suivante:

```
" 1254 LMA 75" NOMIN$ $!   110.30   125.05      45.00 ... 17.55
```

qu'il s'empresse d'exécuter. Ensuite, HISTO passe en mode graphique, trace l'histogramme, l'imprime et redonne la main à dBASE par un simple BYE. Ca vous donne des idées, hein. Et si vous croyez que c'est tiré par les cheveux, eh bien il est plus pratique de tracer un histogramme sous FORTH qu'en dBASE. dBASE est doué pour traiter les données, TURBO-Forth est doué pour tout (pour rappel, RAPIDfile, le petit frère de dBASE, est écrit en FORTH).

$EXECUTE (str ---)
Cette fonction déjà connue de nos lecteurs assidus nous semble tellement importante que nous l'adoptons en tant que mot standard (tant pis pour les autres versions de FORTH). La grande nouveauté de ce mot est d'autoriser la réentrance. Lors de l'exécution d'une chaine, un mot de cette chaîne peut lui-même utiliser $EXECUTE. Ce mécanisme est limité à la capacité de la pile de données. C'est presque le traitement de liste en somme... Sachant que TURBO-Forth peut extraire une sous-chaîne par recherche de chaîne, par découpage ou par sélection d'occurence, vous n'aurez plus aucune peine à créer un mini interpréteur LISP, LOGO ou ...BASIC. Exemple d'utilisation de recherche par occurence, le mot ITEM:

```
255 STRING A$
" 110.30 125.05 45.00 17.55 " A$ $!
" 44.35 135.70 " A$ APPEND$

A$ 1 ITEM TYPE    affiche 110.30
```

Transformation numérique d'un ITEM:

```
A$ 3 ITEM $EXECUTE D.  affiche 4500
```

Mais aussi ce genre de plaisanterie:

```
" DARK WORDS BYE " A$ $!
A$ 1 ITEM $EXECUTE    efface l'écran
A$ 2 ITEM $EXECUTE    affiche le vocabulaire
```

TURBO-Forth lui-même exploite $EXECUTE au démarrage. Ainsi, si depuis MSDOS ou un autre langage (dBASE III/III+, TURBO-Pascal, etc...) vous lancez:

```
TURBO WORDS BYE
```

vous lancez TURBO et exécutez la chaîne qui suit le mot TURBO. Ainsi peut-on exploiter une application entière ou seulement un module. Le démarrage de TURBO seul affiche le menu et initialise les touches de fonctions; le démarrage de TURBO avec passage de paramètres exécute immédiatement la fonction demandée. Le menu n'est pas affiché.

BYE ([c] ---)
Bien connu par vous, ce mot a été réécrit en assembleur. On peut le faire précéder d'un paramètre optionnel. Ce paramètre peut être exploité en sortie de TURBO-Forth par la fonction ERRORLEVEL dans un fichier .BAT par exemple. Pour résumer, TURBO-Forth peut recevoir des données par INCLUDE ou par passage de paramètres au démarrage; TURBO-Forth en renvoie par la fonction SAVE ou BYE.

Le vocabulaire DOS a été supprimé. C'est certes discutable, mais si les primitives telles que (DO) ou (LOOP) n'ont pas été séparées du dictionnaire principal, il n'y a pas de raison de séparer CLOSE et (CLOSE).

A: B: C: D: (---)
Jusqu'à quatre drives peuvent maintenant être sélectionnées:

- 2 lecteurs disquette et un disque virtuel (option VDISK dans CONFIG.SYS).
- 1 lecteur disquette, un disque dur et un disque virtuel (idem).

La table CC-FORTH a été remaniée. Désormais, on ne peut plus compiler de codes de contrôles tel ESCAPE dans une chaîne de caractères. Si dans un fichier source vous mettez des tabulations, elles seront automatiquement converties en espaces (une tabulation= un espace).

REPLICATE (n char ---)
Reproduit n fois le caractère char. Exemple:

```
: BARRE 80 ASCII ░ REPLICATE ;
```

En conséquence, les définitions de SPACES et BACKSPACES deviennent:

```
: SPACES (S n -- ) BL REPLICATE ;
: BACKSPACES   (S n -- ) BS REPLICATE ;
```

Les constantes 0, 1, 2 et 3 ont été réécrites en assembleur:

```
CODE 0   0 # AX MOV   1PUSH   END-CODE
CODE 1   1 # AX MOV   1PUSH   END-CODE
CODE 2   2 # AX MOV   1PUSH   END-CODE
CODE 3   3 # AX MOV   1PUSH   END-CODE
```

de même que pour BS, BL. D'autres primitives ont également été réécrites et optimisées. Deux nouveaux types de données sont déclarées en tête du méta-compilateur, 2CONSTANT et STRING:

```
LABEL DO2CONSTANT  W INC   W INC   0 [BX] AX MOV
      2 [BX] DX MOV   2PUSH   END-CODE
LABEL DOSTRING     W INC   W INC   BX AX MOV   AX  INC
    AX INC   AX PUSH
      1 [W] AX MOV     0 # AH MOV    1PUSH    END-CODE
```

Génération des données dans la cible:

```
: 2CONSTANT     (S dn --)
    RECREATE   [[ ASSEMBLER DO2CONSTANT ]]
    LITERAL ,-T   OVER OVER   ,-T  ,-T  2CONSTANT ;
```

```
  : STRING          (S n --)
     RECREATE  [[ ASSEMBLER DOSTRING ]] LITERAL ,-T
     DUP C,-T  0 C,-T  DUP ALLOT-T  STRING ;
```

Création du type dans le noyau TURBO:

```
  : 2CONSTANT  (S d ---)   CREATE  , ,  ;USES
     DO2CONSTANT ,
  : STRING  ( n --- <nomvar$> en compilation)
     CREATE DUP C, 0 C, DP +!
     ;USES DOSTRING ,
```

Les mots de gestion d'écran tels que DARK et MODE restent vectorisés, mais avec une grosse différence par rapport à la version LAXEN et PERRY:

```
  CODE (AT)   (S col row -- col row)
     AX POP   DX POP   DX PUSH   AX PUSH
     AL DH MOV  BH BH XOR  2 # AH MOV   16 INT   NEXT   C;
  : AT   (S col row -- )  ( 0 0 est le coin gauche )
     NOOP  2DUP #LINE !  #OUT ! ;
  \ initialisé par ' (AT) IS AT
  VARIABLE GRMODE
  CODE (MODE)  ( n ---     n:=[0..7] )
     GRMODE #) AX MOV   16 INT   NEXT  C;
  : DARK   (S -- )
     NOOP   #LINE OFF  #OUT OFF ;
  \ initialisé par ' (MODE) IS DARK
  : MODE ( n ---)
     GRMODE !  DARK  ;
```

Le mot (AT) prend la place de NOOP dans la définition de AT dès que la pseudo-vectorisation est activée par la séquence:

```
  ' (AT) IS AT
```

Par parenthèse, le mot IS permet également de gérer des constantes sous forme de pseudo-constantes:

```
  1987 CONSTANT ANNEE-COURANTE
```

peut devenir dans pas longtemps:

```
  1988 IS ANNEE-COURANTE
  ANNEE-COURANTE .   affiche 1988
```

Le décompilateur tient compte des nouveaux types de données 2CONSTANT et STRING. Exemple:

```
  30 STRING A$    " pour exemple" A$ $!
  SEE A$
affiche
  30 STRING A$   contenu:pour exemple:
```

Les fonctions de traitement de chaînes sont intégrées à TURBO-Forth. Des fonctions d'accès mémoire et de transfert de blocs inter-segments vous donnent accès à TOUTE LA MEMOIRE VIVE!!!

DIFFUSION DE TURBO-Forth

TURBO-Forth est diffusé sous deux formes:

- une version d'évaluation, disponible contre 10 timbres à 3,70Fr (càd les timbres pour expédition de 10 numéros de JEDI), contenant le seul fichier TURBO.COM. Cette version n'est ni bridée, ni limitée, ni plombée, ni piégée. Elle est disponible de suite.

- une version développeur, comprenant le méta-générateur, les fichiers source de TURBO-Forth, l'éditeur plein écran, la documentation et des fichiers d'exemples. La documentation est en cours de composition. Cette version ne sera pas disponible avant mars/avril 1988. Nous sommes preneurs des extensions suivantes:
  - exécution des bibliothèques FORTRAN
  - exécution des bibliothèques PASCAL
  - accès aux fichiers dBASE III/III+
  - accès aux données MULTIPLAN
  - gestion graphique HERCULES
  - gestion coprocesseur math.8087
  - transmission TELECOPIE SAGEM (1200 bauds)

et tout ce qui vous passera par la tête.

Faites parvenir vos disquettes et participez ainsi au succès de TURBO-Forth 83-Standard. Et que vive le Forth.

REGLES DE TYPOGRAPHIE

par M. PETREMANN

Au vu des différents manuscrits et articles dactylographiés que nous recevons régulièrement, il nous a semblé urgent de vous donner quelques rudiments de typographie. Disposer d'un bon traitement de texte n'est pas l'essentiel si votre prose doit quand même être re saisie par votre imprimeur. Si nous même ne sommes pas très regardant sur vos petites erreurs, ce serait bien plus gênant le jour où vous présenterez un document "clef en main" de 100 à 300 pages à un imprimeur très pointilleux. Les conseils diffusés ici vous feront peut-être gagner un jour un temps précieux.

PREMIERE REGLE, LA MISE EN PAGE

Votre texte doit être parfaitement centré. Une marge suffisante doit être réservée pour l'assemblage des pages à droite et à gauche. Réserver au minimum un centimètre en haut et en bas de page pour la prise en pince offset.

Si votre traitement de texte le permet, cadrez toujours votre texte à droite et à gauche, même pour un courrier. Votre correspondant comprendra ainsi que vous disposez d'un matériel performant et que vous savez vous en servir.

L'indentation de la première ligne de texte d'un paragraphe ou alinéa est à réserver pour votre courrier "à la française". Ouvrez un livre, n'importe lequel, et regardez si on indente la première ligne,... vu?!

LA NUMEROTATION DES CHAPITRES

Le système de numérotation est laissé à la discrétion du rédacteur. Cependant, quelques règles prédominent:

1) la numérotation ISO. On découpe les chapitres en sous-chapitres repérés par des chiffres arabes séparés par un point. Exemple:

```
1.
 1.1.  ... 1.2.   ...  1.3.
2.
 2.1.
   2.1.1.   etc...
```

2) la numérotation typo:

```
I.
  A.
    1.
      a.
        (1)
II.  etc...
```

On évite de définir plus de cinq niveaux de numérotation. Dans ce cas, l'indentation du sous-chapitre est acceptée. Selon le traitement de texte utilisé, on peut mettre en valeur le numéro du chapitre:

1) jhgfkh jjkhgfkj ljhjh è oitot upiyp nbnb,n ljk jjg hjh gh jj jhjh jhjhg jjg.   << est toléré.

```
  1)   jhgfkh jjkhgfkj ljhjh è oitot  upiyp nbnb,n  ljk
       jjg  hjh gh jj jhjh jhjhg jjg.   << est souhaitable.
```

NUMEROTATION DES PAGES

Une page est TOUJOURS numérotée avec le numéro impair à droite, le numéro pair à gauche. Exceptionnellement, le

numéro peut être placé au milieu du bas de page. Si votre document sera diffusé par photocopie recto seulement, les numéros serons placés exclusivement à droite.
Un cartouche de haut ou de bas de page peut rappeler le libellé du chapitre ou du document.

PLACE DE LA PONCTUATION

Les points, les points-virgule, points d'interrogation-exclamation, points de suspension, double-points sont TOUJOURS accolés au mot qui précède.

EXEMPLE: << OUI
EXEMPLE : << NON

C'est pourtant simple, imaginez que vous écriviez un rajout en plein paragraphe, votre EXEMPLE séparé des deux-points deviendrait en fin de ligne:

... ligne de texte... EXEMPLE
:

ce serait moche.

Une parenthèse ouvrante est toujours accolée au mot qui suit, une parenthèse fermante au mot qui précède:

(Ndlr: jhfjhgf) << OUI
(Ddlr: jhkjh kjh) << NON

Il en est de même pour les crochets et les accolades (normalement inutilisés en typographie dans un texte à caractère littéraire).

LES VEUVES ET LES ORPHELINS

Tout paragraphe de plus de 3 lignes situé en bout de page doit autant que possible ne pas être séparé de sa ligne de fin, idem en début de page suivante pour la première ligne de paragraphe. Ces lignes isolées sont appelées veuves et orphelins. Si un paragraphe doit se poursuivre sur une ligne suivante, on passera au moins les deux dernières lignes sur la page suivante. Il en est de même pour un début de chapitre.

Dans le cas de citations ou d'exemples, on verrouille la ligne faisant référence à la citation concernée en un bloc.

.... pour exemple:] verrouillage du bloc
jhgfjh jghffj

ceci bien entendu si votre traitement de texte dispose de la fonction de verrouillage de bloc.

LES CESURES DE MOTS

Un mot est toujours coupé en fin de ligne entre deux syl-la- bes. Pour rappel, une syllabe se termine généralement après une voyelle accentuée ou entre deux lettres jumelées, après une voyelle autre que E suivie d'une consonne. Certaines consonnes regroupées à la voyelles qui précède et formant ainsi un groupe phonétique (EN ON UN OIN...) sont coupés après la consonne:

é- gli- se
ex- cla- ma- tion
ta- bleau
car- ré
con- son- ne
etc...

On évite autant que possible de laisser une lettre isolée en fin de ligne: on évitera é- glise au profit de égli- se ou église. Le cas particulier de la lettre x est équivalent au groupe cc (c'est pour cette raison que le E de "exemple" n'est pas accentué").

LES ABREVIATIONS:

En dehors des terminologies scientifiques et techniques particulières, sont admisés les abréviations suivantes:

ex: pour exemple (et non par exemple)
n° numéro
etc et caetera
Ndlr note de la rédaction (réservé à la rédaction)

En respectant ces quelques règles, vous ferez gagner un temps précieux à ceux qui exploiterons votre prose. Un texte bien tourné est apprécié, mais bien tapé il évitera à votre éditeur l'éruption de boutons purulents provoquée par les coquilles défraichies.

ACCES AUX COMMANDES DOS

par Y. SURREL

Cher Secrétaire

Comme promis je vous envoie le listing des mots permettant d'avoir accès aux commandes DOS à partir du FORTH. La syntaxe est très simple; pour exemple:

```
!" DIR A:|SORT|MORE>PRN"
```

en mode direct, ou:

```
: NICEDIR !" DIR A:|SORT|MORE>PRN" ;
```

en mode compilé; à noter que:

```
!" COMMAND"
```

renvoie dans l'environnement DOS, avec l'affichage du prompt (ex: A:\>), comme si l'on était sorti du FORTH. On revient à FORTH en faisant:

```
A:\>EXIT
```

et l'on récupère notre bon "ok" forthien.

J'espère que ce programme vous intéressera. Amicalement.

Y. SURREL

LISTING:

```
( Commande SHELL pour DOS ) v 2.0 - Y. SURREL / 2-12-87 )
( La manière d'appeler le processeur de commande est ex )
( pliquée dans "DOS version 3.00 TECHNICAL REFERENCE"   )
( pp 7-3 )

ONLY FORTH ALSO DEFINITIONS

HEX
CODE SETBLOCK (S -- )
     BX PUSH
     1000 # BX MOV
     4A # AH MOV
     21 INT
     FFFF # BX MOV
     48 # AH MOV
     21 INT
     BX POP
     NEXT C;

CREATE NOMFICH
     ," C:\COMMAND.COM" 0 ,

CREATE PARAMBLOCK
     0 , 0 , 0 , 5C , 0 , 6C , 0 ,

CODE EXEC (S -- )
     BX PUSH
     BP PUSH
     SI PUSH
     CS AX MOV
     AX PARAMBLOCK 4 + #) MOV
     AX PARAMBLOCK 8 + #) MOV
     AX PARAMBLOCK 2+ 8 + #) MOV
     4B00 # AX MOV
```

```
     NOMFICH 1+ # DX MOV
     PARAMBLOCK # BX MOV
     21 INT
     SI POP
     BP POP
     BX POP
     NEXT C;
DECIMAL

: (!")                      (S add len -- )
    TUCK                    (S len add len )
    PAD 3 + PLACE           (S len )
    " /C " PAD PLACE        (S len )
    3 + DUP PAD C!          (S len' )
    1+ PAD + 13 SWAP C!     ( CR au bout de la chaîne )
    PAD PARAMBLOCK 2+ !
              ( initialise  2ème mot de PARAMBLOCK )
    EXEC ;

: !"                        (S -- )
    SETBLOCK
    STATE @
    IF
      [COMPILE] "
      COMPILE (!")
    ELSE
      ASCII " PARSE         (S add len )
      (!")
   THEN ; IMMEDIATE

( Commande SHELL pour DOS ) v 2.0 )
ONLY FORTH ALSO DEFINITIONS
HEX
CODE SETBLOCK (S -- )
     BX PUSH
     1000 # BX MOV
     4A # AH MOV
     21 INT
     FFFF # BX MOV
     48 # AH MOV
     21 INT
     BX POP
     NEXT C;

CREATE NOMFICH
     ," C:\COMMAND.COM" 0 ,

CREATE PARAMBLOCK
     0 , 0 , 0 , 5C , 0 , 6C , 0

CODE EXEC (S -- )
     BX PUSH
     BP PUSH
     SI PUSH
     CS AX MOV
     AX PARAMBLOCK 4 + #) MOV
     AX PARAMBLOCK 8 + #) MOV
     AX PARAMBLOCK 2 + 8 + MOV
     4B00 # AX MOV
     NOMFICH 1+ # DX MOV
     PARAMBLOCK # BX MOV
     21 INT
     SI POP
     BP POP
     BX POP
     NEXT C;
DECIMAL

: (!")                      (S add len -- )
    TUCK                    (S len add len )
    PAD 3 + PLACE           (S len )
    " /C " PAD PLACE        (S len )
    3 + DUP PAD C!          (S len' )
    1+ PAD + 13 SWAP C!     ( CR au bout de la chaîne )
    PAD PARAMBLOCK 2+ !
              ( initialise  2ème mot de PARAMBLOCK )
    EXEC ;

: !"                        (S -- )
    SETBLOCK
    STATE @
    IF
      [COMPILE] "
      COMPILE (!")
    ELSE
      ASCII " PARSE         (S add len )
      (!")
   THEN ; IMMEDIATE
  QUIT
```

Ndlr: sincères remerciements de la part des lecteurs de JEDI pour votre très intéressante routine. A noter que dans TURBO-Forth, une variante a été implantée, nommée SHELL:

```
" string" SHELL
```

exemple:

```
" DIR *.COM" SHELL
```

LES NOUVEAUX LANGAGES ET L'EXEMPLE D'HYPERTALK

par J.PERRET

Périodiquement apparaissent de "nouveaux langages" de programmation qui soit se présentent comme des extensions d'anciens (FORTH ou C ou PASCAL "orienté objet"), soit parraissent totalement nouveaux (PROLOG?). L'amateur de langages se précipite sur sa littérature favorite et y découvre des choses pouvant aller de quelques copies d'écran avec la liste des instructions à des articles plus sérieux illustrés d'exemples. Il y manque souvent le principal, à savoir:

- qu'est-ce que ce langage apporte de neuf, c'est à dire quels sont les nouveaux concepts?

Un nouveau langage doit inciter son utilisateur à les utiliser (les concepts) et l'empêcher (doucement ou fermement) de faire du BASIC en PASCAL, du PASCAL ou du FORTRAN en LISP, du C en C++, etc... qui est la pente naturelle de tout programmeur.

Je voudrais illustrer celà à l'aide du cas d'Hypercard, un générateur de programmes muni d'un langage orienté objet appelé Hypertalk et destiné aux heureux possesseurs de MacIntosh (MacPlus et supérieurs).

Contrairement à ce que j'ai lu, on peut très bien faire des essais avec un MacPlus et un seul lecteur. Evidemment, un disque dur est fort utile mais pas indispensable pour faire quelques essais.

Un langage orienté objet permet de regrouper dans une même entité (l'objet) des données et des procédures ou méthodes. Des mesages sont échangés entre les objets et l'utilisateur. Chaque objet s'occupe des messages qui le concernent (pour lequel il possède une méthode). A tout moment on peut rajouter un objet obtenu par copie et modification éventuelle d'un objet. Le premier concept de la programmation "objet" me semble être un anti-concept: disparition de la séquence (adieu les numéros de ligne). Au fait, les tableurs ne seraient-ils pas les ancêtres des L.O.O? (Exercice mental: ressortez votre vieux VISICAL de la poussière et regardez-le comme un L.O.O).

Mais il y a mieux: si un objet ne sait pas quoi faire d'un mesage, il le passe à son père (ou à sa mère, je ne sais pas) ce qui permet une hiérarchie des traitements.

Hypercard permet de manipuler (au moins) 4 sortes d'objet: des piles qui contiennent des boutons, et des champs. Chaque objet possède certaines caractéristiques propres et un "script", sorte de scénario décrivant dans le langage Hypertalk les réponses à donner aux messages. Les messages peuvent être écrits par l'utilisateur dans une "boite à message" ou déclenchés par diverses actions (l'appui sur un bouton par exemple). L'emploi de la souris en est très facilité.

Prenons l'exemple d'un chirurgien désirant se faire un fichier sur différents types d'intervention. Il crée une pile "interventions" par simple clonage de la pile-mère (appelée Home-stack) qu'il orne d'un superbe dessin représentant un(e) patient(e) (des accessoires de dessin sont fournis mais on peut aussi récupérer des images provenant d'autres applications). Il se fabrique ensuite un modèle de fiche "intervention" avec des champs pour les commentaires, les mots clés, une zone de dessin, tout celà très simplement en choisissant dans un menu. Il y enregistre sa dernière opération du genou qu'il baptise "genou" (tout objet créé se voit automatiquement affecté d'un numéro d'identification, mais on peut aussi lui donner un nom). Si certains champs, dessins, boutons doivent être communs à d'autres cartes, on les crée sur une "carte de fond" (background), la carte ne contenant que les données spécifiques. Jusqu'ici, celà ressemble fort à l'utilisation d'un logiciel de création de fichiers en beaucoup plus souple. Il revient ensuite à sa première fiche (celle du dessin du patient) et crée un bouton invisible qu'il place "sous" le genou. Le script de ce bouton sera:

```
on mouseUp
go to card "genou1"
end mouseUp
```

Ce qui se traduit à peu près par:

à la rception du mesage "bouton de la souris appuyé puis relaché" aller à la carte...

C'est aussi simple! Tout est automatiquement enregistré et opérationnel, on peut essayer immédiatement:

on clique sur le genou et hop! la fiche genou apparait!!

Il reste à améliorer la carte "fond" en ajoutant des boutons permettant de créer une nouvelle carte, de passer à la suivante, la précédente, la première, de trouver une fiche par mot clé. Chaque bouton peut être illustré d'un dessin ou d'une icone rendant sa fonction plus évidente qu'un long baration.

Ce genre d'exercice de création prend environ une heure. En fait, il peut prendre beaucoup plus, car on s'amuse à ajouter des boutons plus ou moins utiles, à illustrer la carte, etc... On peut copier une carte, un bouton, un fond, un dessin d'une autre pile avec ses caractéristiques et son script, puis les éditer. Tout celà sans (ou presque sans) taper la moindre ligne, mais en dessinant, déformant, en choisissant des options dans des menus "à la Mac". Il est possible d'obtenir facilement une "interface utilisateur" très simple. Un bouton illustré d'une main jetant une fiche à la poubelle se passe totalement d'explications.

Peut-on faire autre chose que des agendas, des fichiers et des mémos?

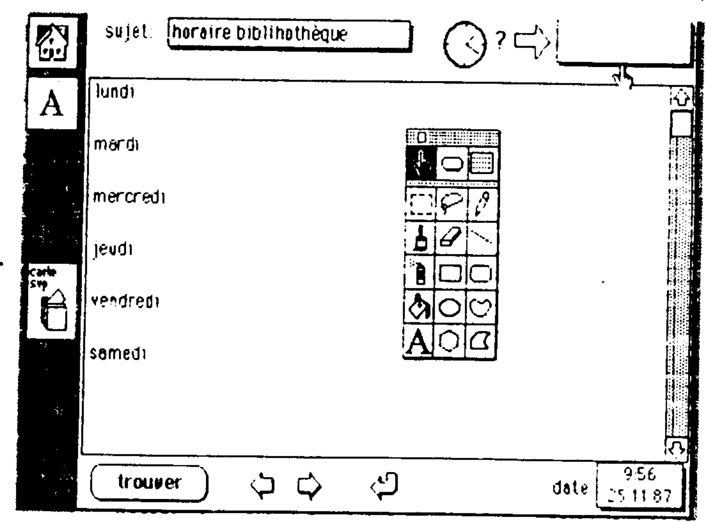
On peut très facilement écrire des petits didacticiels (quel horrible mot) ou des mini-tableurs spécialisés. On peut s'écrire un lanceur d'applications personnalisé à son Finder.

Peut-on écrire un système expert? Un truc-assisté par ordinateur? Quelque chose de carrément nouveau (et de préférence utile)? Je cherche... Si vous avez trouvé...

Réfrences: bibliothèque HyperCard, cité Mac sur Calvacom. Un exemple de carte. Au milieu la palette des instruments avec en haut et de gauche à droite:

la main à tester les boutons
l'outil à manipuler les boutons
l'outil à faire les champs

l'outil à manipuler les boutons
l'outil à faire les champs



COURRIER: GEM et FORTH

Je cherche ces renseignements pour utiliser GEM avec FORTH sur mon PC ainsi que la mise en place d'un FORTH multi-tâches (n mots s'exécutant en parrallèle; implémentation de sémaphores)

Gilles CASTET

Réponse: très bonne question et nous la posons également. Y-a-t il un adhérent qui planche sur le problème?...

COURRIER: SAUCISSONAGE

Evitez le renvoi d'une fin d'article enclave dans un autre article à une autre page dans un autre contexte (saucissonage). La revue Sélection du Readers Digets procède ainsi. C'est HORRIPILANT!!. Exemple de mise en page: ELEKTOR, 3 ou 5 photocopies et l'article est complet.

Tentative de communication par messagerie Minitel:

3613: trop cher pour l'exploitant
3615: trop cher pour l'usager (interdit aux PTT)
xx.xx.xx.xx micrserveur mono-accès, pas de dialogue, risque d'encombrement.
3614: libre service pour les agents des PTT
peu cher, 0,37 Fr à tarif rouge
0,19 Fr à tarif bleu
0,13 Fr à tarif bleu nuit
tarif dégressif du téléphone.

Boite aux lettres 3614 code RTEL. Tout courrier sera retransmis au secrétaire/président.

Mon avis: l'hébergement sur serveur est dangeureux. L'hébergé est lié par contrat au serveur qui se réserve le droit d'utiliser le nom en cas de défaillance de l'hébergé, exemple H6 créé par HEBDOGICIEL, tombé entre les mains de CANAL 4.

Erik FORGET
92320 CHATILLON SOUS BAGNEUX

Réponse: le renvoi d'une fin d'article est parfois chose nécessaire si on ne veut pas gaspiller de la place. Sans le saucissonage, JEDI devrait compter 2 ou 3 pages de plus. Ceci dit, nous ne le faisons qu'en dernière extrémité.

suite page 10

Cher secrétaire,

La lecture du n°38 et de votre réponse à mon courrier y incluse m'a rassuré. Et je souhaite longue vie à "JEDI sauvé de l'oubli". Vous trouverez ci-joint mon réabonnement en retour.

Sur un autre plan, votre idée d'un "Turbo-Forth" est très séduisante. Mais je pleure des larmes de sang de ne disposer que d'un 6809 qui ne digère pas l'assembleur 8086/88. Ne pourriez-vous transposer celui-ci en Forth? ou en assembleur Forth/6809? ou en assembleur Forth/68000 (bien sûr très voisins)? ou, enfin, donner une définition algorithmique des 3 mots (OPEN-F) (CLOSE-F) F-GET telle que les "Non-PC" puissent bénéficier de votre ingéniosité bien connue sans être obligé pour cela... d'acheter de l'asiatique!

Je vous rappelle à ce propos qu'un nombre non négligeable d'adhérents Jedi dispose de 6809 ou de 68000. Et à ce titre, si l'on veut fixer tout ce monde là, ne pourrait-on tenter de donner dans chaque cas où l'assembleur est indispensable, pour des raisons de rapidité, soit une version du mot dans chaque grand assembleur, soit une version Forth , à charge pour chacun de retransposer dans son assembleur personnel? Je pose la question et promets, pour ma part, de le faire dans les petits trucs que je vous enverrai.

Bien sûr, les génies transposent sans difficulté, mais moi... j'en suis incapable, ne connaissant (et encore très mal) que l'assembleur Forth-6809.

Amicalement.

Bernard C. LAMBEY
34070 MONTPELLIER

Réponse:

En Forth deux écoles s'affrontent: ceux qui veulent écrire le maximum en FORTH (certains sont allés jusqu'au LIT en Forth); ceux qui veulent un rendement maximum, donc un Forth tout langage machine (cas du Forth CANON X07). Pour ma part, je me situe entre les deux, n'utilisant l'assembleur que par nécessité. Et dans le cas de INCLUDE diffusé dans le n°38, les mots (OPEN-F) (CLOSE-F) et F-GET me semblent assez explicite pour ne pas vous embarrasser d'une littérature trop abondande:

(OPEN-F) ouvre un fichier MSDOS. Pour vous, sous FLEX, il faut connaître la routine à activer au niveau du DOS pour remplir la même fonction. Si votre FORTH dispose du mot CALL, taper adr CALL en empilant ou en chargeant au préalable les registres ou les adresses avec les paramètres adéquats (adresse de la chaine de descipteur de fichier...)

(CLOSE-F) ferme le fichier. Sans autre commentaire.

F-GET est équivalent à KEY, à ceci près qu'il lit un caractère depuis le fichier précédemment ouvert et non depuis le clavier. En outre, la revectorisation de KEY sur cette nouvelle fonction introduit les caractères saisis directement dans le TIB.

Connaissant le FORTH 6809 sur THOMSON, je puis affirmer que toutes les fonctions équivalentes sont déjà implantées, reste à revectoriser judicieusement KEY.

Mais plus qu'un simple problème de traduction FORTH/assembleur et inversement, c'est le problème du DOS et de la gestion mémoire qui se pose. En effet, Turbo-Forth peut ouvrir plusieurs fichiers en même temps à condition de sauvegarder les paramètres (voir HANDLE) de ces fichiers, ce qui ne peut être réalisé de manière identique sous FLEX, CP/M ou DOS-FORTH-THOMSON. La puissance de Turbo-Forth est dûe en partie aux caractéristiques de MSDOS. Conséquence: Turbo-Forth ne peut être disponible que pour les systèmes IBM et compatibles. Si un programmeur courageux l'adapte à un autre système (ATARI par exemple), le choix des primitives différera et peut-être aussi les caractéristiques. L'exploitation des ressources du système est à ce prix et est aussi une des raisons de la difficulté à conserver un standard.

TURBO-Forth reste standard, c'est à dire que le comportement de toutes les primitives définies par le FORML n'a pas été modifié. Exemple:

```
n SPACES affiche n espaces
```

pourtant, la définition de SPACES dans Turbo-Forth a été modifiée:

```
: SPACES BL REPLICATE ;
```

car le standard ne précise pas qu'une définition de mot standard doive être écrite avec des mots standards. Voilà, et s'il vous prenait l'envie de nous envoyer un programme pour 6809 ou 68000 intraduisible en Turbo-FORTH, nous ne nous opposerons pas à sa diffusion (surtout s'il est génial...).

M. PETREMANN

FONCTION HORLOGE ET CALENDRIER

par Yves SURREL

pour TURBO-Forth
F83 Laxen et Perry MSDOS

Cher Secrétaire:

Voici un certain nombre de mots pour F83 standard permettant de gérer la fonction horloge des PCs. Il y a d'abord les primitives (TIME) et (DATE) en code machine qui appellent les interruptions 2Ch et 2Ah.

TIME laisse sur la pile un nombre double précision égal au nombre de centièmes de secondes écoulées depuis zéro heure. HH:MM:SS,CC reprend ce nombre et laisse l'adresse et la longueur d'une chaîne de caractères formatée.

TIME$ couple les deux mots précédents, et .TIME affiche l'heure. Le mot .INTERVAL prend deux nombres double précision d1 (début) et d2 (fin) du type laissé par TIME et affiche la durée séparant d1 et d2. Exemple d'utilisation:

```
:: TIME 60000 0 DO LOOP TIME .INTERVAL ;
```

On obtient 00:00:01,04 avec un AMSTRAD PC1512.

DATE$ laisse l'adresse et la longueur d'une chaîne de caractères contenant la date, et .DATE l'affiche.

Enfin, je vous propose le mot $)SCREEN qui affiche directement à l'écran une chaine de caractères avec un attribut contenu dans la variable ATTR.

Selon sa configuration vidéo, il faudra mettre $H000 au lieu de $B800 dans VIDEO_RAM. Ce mot me sert pour définir la tâche de fond TICTAC qui affiche l'heure en permanence; n'oublions pas que le FORTH est multitâche! A noter que:

```
BACKGROUND: TICTAC BEGIN 25 0 AT .TIME AGAIN ;
```

plante la machine...

Pour lancer TICTAC, faire MULTI puis TICTAC WAKE.

Amitiés.

LE LISTING:

```
\ Programme pour afficher la date et l'heure
\ (C) Yves SURREL Décembre 1987

ONLY FORTH ALSO DEFINITIONS

HEX

CODE (TIME) (S -- hhmm sscc )
```

```
    2C # AH MOV   21 INT
    CX PUSH     \ ch: heure; cl: minutes
    DX PUSH     \ dh: secondes; dl: 1/100s
    NEXT C;

CODE (DATE) (S -- aaaa mmqq xx0j )
    2A # AH MOV   21 INT
    CX PUSH     \ cx: année
    DX PUSH      \ dh: mois; dl: quantième
    AL PUSH      \ al: jour de la semaine
    NEXT C;

DECIMAL

: TIME (S -- d ) \ durée depuis minuit en 1/100s
  (TIME)            (S hhmm sscc )
  256 /MOD          (S hhmm cc ss )
  100 * +           (S hhmm cc )
  S>D               (S hhmm dcc )
  ROT 256 /MOD      (S dcc mm hh )
  60 * +            (S dcc mm )
  6000 UM*          (S dcc dcc )
  D+  ;

: HH:MM:SS,CC     (S d -- add len )
  \ formattage de d en centièmes
  BASE @ >R
  <# DECIMAL # #    \ Centièmes
  ASCII , HOLD
  # 6 BASE ! #      \ Secondes
  ASCII : HOLD
  DECIMAL # 6 BASE ! #  \ Minutes
  ASCII : HOLD
  DECIMAL # #            \ Heures
  #>
  R> BASE !  ;

: TIME$ (S -- add len )
  TIME HH:MM:SS,CC  ;

: .TIME (S -- )
  TIME$ TYPE  ;

: .INTERVAL       (S d1 d2 -- )
  2SWAP D- HH:MM:SS,CC TYPE  ;

CREATE 'DATE 11 ALLOT
   ASCII - C, 2 ALLOT
   ASCII - C, 4 ALLOT

: ##>MEM (S n add -- add+3 )
  \ Convertit n en 2 digits à l'adresse add
  SWAP 0 <# # # #> 2 PICK SWAP CMOVE 3 +  ;

: DATE$ (S -- add len )
  BASE @ >R DECIMAL
  (DATE)
  15 AND
  CASE 0 OF " Dimanche " ENDOF
       1 OF " Lundi "    ENDOF
       2 OF " Mardi "    ENDOF
       3 OF " Mercredi " ENDOF
       4 OF " Jeudi "    ENDOF
       5 OF " Vendredi " ENDOF
       6 OF " Samedi "   ENDOF
  ENDCASE
  'DATE DUP 9 BLANK SWAP CMOVE  \ Jour de la semaine
  256 /MOD SWAP [ 'DATE 9 + ] LITERAL
  ##>MEM ##>MEM \ Quantième
  >R
  0 <# #S #> R> SWAP CMOVE   \ Année
  R> BASE !
  'DATE 19 ;

: .DATE (S -- )
  DATE$ TYPE  ;

VARIABLE VIDEO_RAM
  HEX     B800 VIDEO_RAM !     DECIMAL

VARIABLE ATTR  2 ATTR !
  \ Attribut d'affichage: encre verte

CODE $>SCREEN (S add len n -- )
  \ Affiche la chaine de caracteres définie par add et len
  \ à la position n de l'écran
     DI POP        \ destination...
     DI SHL        \ ... 2* à cause des attributs
     CX POP        \ nb d'octets à afficher
     AX POP        \ source...
     SI PUSH       \ ... mais il faut sauvegarder ...
     AX SI MOV           \ ... SI
     VIDEO_RAM #) AX MOV \ segment destination...
     AX ES MOV           \ ... dans ES
     ATTR #) AX MOV      \ attribut d'affichage dans AX
     CLD
   ( DO ) HERE
      BYTE MOVS          \ SI et DI s'incrémentent de 1
      AL STOS            \ DI s'incrémente de 1
     LOOP
     SI POP              \ restauration de SI...
     DS AX MOV           \   ... et...
     AX ES MOV           \  ... de ES
     NEXT C;

BACKGROUND: TICTAC
   BEGIN PAUSE TIME$ 69 $>SCREEN AGAIN ;
```

RECREATIONS APL
LES TOURS DE HANOI

par F. ESPINASSE

Le problème des Tours de Hanoï est une amusante application de la récursivité pour les langages supportant cette fonctionalité. Citons les plus connus par ordre alphabétique: APL, les BASICs structurés (QuickBASIC, TrueBASIC), FORTH, MODULA-2, PASCAL, PL1. Un grand absent: FORTRAN (Ndlr: ah, bon! et COBOL?...).

Le principe du jeu est bien connu: on dispose de trois socles A, B, C. Sur l'un d'eux sont empilés un certain nombre de disques de diamètres décroissants de bas en haut. On doit déplacer la pile sur un autre socle en procédant disque par disque et en utilisant le 3ème socle comme intermédiaire. A aucun moment un disque quelconque ne doit supporter un disque d'un diamètre supérieur au sien.

L'ALGORITHME

La solution se trouve assez simplement en remarquant que pour déplacer une pile de N disque de A en B, il suffit (et il faut!):

1) déplacer une pile de N-1 disques de A en C

2) déplacer le disque N de A en B

3) déplacer une pile de n-1 disques de C en B

Ces trois étapes sont obligatoires. On saura donc déplacer une pile de N disques si l'on sait déplacer une pile de N-1 disques. On écrira donc une fonction DEPLACE, qui par deux fois s'appelle elle-même. Chaque fonction appelée s'appelle elle-même deux fois, etc...

```
[  0] N DEPLACE A;⎕IO
[  1] ⎕IO←1
[  2] A←'ABC'⍳A
[  3] →(0=N)/0
[  4] →(0=∧/A∊ 1 2 3)/ERR
[  5] →(∨/2≠⍴A)/ERR
[  6] (N-1)DEPLACE  'ABC'[(1↑A),6-+/A]
[  7] '  DEPLACER LE DISQUE  ',(1 0 ⍕N),'  DE  ',('ABC'[1↑A]),'  EN  ','ABC'[¯1↑A]
[  8] (N-1)DEPLACE  'ABC'[(6-+/A),¯1↑A]
[  9] →0
[ 10] ERR:'DEPLACE: erreur de données'
```

On est chaque fois ramené au problème précédent... mais avec un disque de moins. De proche en proche, on arrive au déplacement d'une pile de zéro disques, qui ne pose plus de gros problème de manutention. Ce cas est traité dans la ligne [3] qui fait tout simplement sortir de la fonction

appelée.

Quelques mots sur la fonction DEPLACE: l'appel de la fonction pour faire passer une pile de 3 disques de B en C s'écrira

3 DEPLACE 'BC'

L'argument gauche de la fonction doit être un nombre entier et l'argument droit une chaine de 2 caractères choisis dans A, B,C. La ligne [2] range dans la variable A les indices des socles de départ et d'arrivée qui doivent être éléments de [1,2,3], ce qui est vérifié à la ligne [4]. La ligne [5] vérifie que ces indices sont bien au nombre de 2. En cas d'erreur on se branche à la ligne [10]. Les lignes [6], [7], [8] sont les trois lignes "opérationnelles" de la fonction qui réalisent les trois étapes décrites plus haut et effectuent les appels récursifs.

Déroulement de la fonction DEPLACE:

```
     3 DEPLACE  'BC'
  DEPLACER LE DISQUE  1  DE  B  EN  C
  DEPLACER LE DISQUE  2  DE  B  EN  A
  DEPLACER LE DISQUE  1  DE  C  EN  A
  DEPLACER LE DISQUE  3  DE  B  EN  C
  DEPLACER LE DISQUE  1  DE  A  EN  B
  DEPLACER LE DISQUE  2  DE  A  EN  C
  DEPLACER LE DISQUE  1  DE  B  EN  C
```

La taille des piles en nombre de disques n'est théoriquement pas limitée. Pratiquement, comme nous allons le voir, le nombre d'opérations élémentaires croit exponentiellement avec le nombre de disques. Le nombre d'appels récursifs croît dans les mêmes proportions, ainsi que la mémoire vive occupée. Il y aura donc nécessairement des limitations physiques.

Un aspect intéressant de notre raisonnement algorithmique est qu'il permet de montrer que la solution décrite est optimum en nombre de déplacements élémentaires et de calculer ce nombre. En effet, noous saurons déplacer de manière optimale une pile de N disques si nous savons déplacer de manière optimale une pile de N-1 disques, car la procédure décrite par les lignes [6], [7], [8] passe par des états intermédiaires OBLIGATOIRES, et il n'en existe pas de plus économique. De proche ne proche, et comme nous connaissons la solution optimale pour déplacer zéro disques... CQFD.

D'autre part, si on appelle Dn le nombre de déplacements élémentaires pour déplacer une pile de N disques, nous avons la relation récursive:

$D_n = 1 + 2.D_{n-1}$

qu'on peut encore écrire:

$[1+D_n] = 2.[1+D_{n-1}] = 2^2.[1+D_{n-2}] = \ldots$

$= 2n.[1+D_0] = 2n$

D'où $D_n = 2n-1$

LA PRESENTATION

La fonction DEPLACE donne une solution rigoureuse du problème des tours de Hanoï, mais si après avoir satisfait votre curiosité intellectuelle vous voulez aussi amuser vos enfants, il faudra lui donner un peu plus de convivialité.

Pour celà nous écrirons une fonction que nous appellerons HANOI (pourquoi pas) qui fera la saisie interactive des données du problème et qui fera ensuite appel à DEPLACE. Nous modifierons un peu cette dernière pour lui faire compter et afficher le nombre d'opérations élémentaires, et pour présenter à chaque pas l'état des trois tours, grâce à l'appel d'une fonction de présentation appelée TOURS.

Voici les listings de ces différentes fonctions, avec un aperçu de la présentation des tours. Bonne soirée.

F. ESPINASSE

```
[  0] HANOI;⎕IO;A;D;NO;T1;T2;T3;I;L;LL;M;NC
[  1] ⍝ Fonction appelée: DEPLACE
[  2] ⍝⎕IO←1+NC←0
[  3] 'PROBLEME DES TOURS DE HANNOI'
[  4] 'TROIS TOURS  A  B  et  C'
[  5] ⍞←¯25↑'TOUR DE DEPART: '
[  6] →(~(D←1↑25↓⍞)∊'ABC')/⎕LC-1
[  7] ⍞←¯25↑'TOUR D''ARRIVEE: '
[  8] →(~(A←1↑25↓⍞)∊'ABC')/⎕LC-1
[  9] →(A=D)/⎕LC-2
[ 10] RE:⍞←¯25↑'NOMBRE DE DISQUES: '
[ 11] NO←1↑⍎25↓⍞
[ 12] →(NO<10)/OK
[ 13] 'NOMBRE DE DISQUES TROP GRAND'
[ 14] →RE
[ 15] OK:''
[ 16] T1←T2←T3←⍳0
[ 17] ⍎'T',(⍕'ABC'⍳D),'←⍳NO'
[ 18] M←(NO,3)⍴0
[ 19] ⍎'M[;',(⍕'ABC'⍳D),']←⍳NO'
[ 20] I←1
[ 21] DEB:L←'    ',(9↑⍕LL),LL←(1 0 ⍕M[I;1]),(M[I;1]⍴⎕AV[221]),((9-M[I;1])⍴' ')
[ 22] L←L,'    ',(9↑⍕LL),LL←(1 0 ⍕M[I;2]),(M[I;2]⍴⎕AV[221]),((9-M[I;2])⍴' ')
[ 23] L←L,'    ',(9↑⍕LL),LL←(1 0 ⍕M[I;3]),(M[I;3]⍴⎕AV[221]),((9-M[I;3])⍴' ')
[ 24] L
[ 25] →((1↑⍴M)≥I←I+1)/DEB
[ 26] '    ',63⍴⎕AV[206]
[ 27] (12⍴' '),'A',(21⍴' '),'B',(21⍴' '),'C'
[ 28] ⍝ (18⍴ 0 0 0 0 0 1)\⍉⌽(3,NO)⍴(NO↑ 1 0 ⍕⌽T1),(NO↑ 1 0 ⍕⌽T2),NO↑ 1 0 ⍕⌽T3
[ 29] ⍝ '    ---    ---    ---'
[ 30] ⍝ '     A      B      C'
[ 31] ''
[ 32] ' Presser une touche pour chaque déplacement'
[ 33] ''
[ 34] NO DEPLACE D,A
[ 35] ' ET VOILA !'
```

Pour information, le saucissonage des articles dans les magazines est un procédé exploité pour forcer les lecteurs à se farcir la publicité (ACTUEL, le FIGARO MAGAZINE, etc....).

Concernant le serveur RTEL, une précédente expérience avec JEDI sur SAM (toujours opérationnel...), nous laisse sceptique. Le système de boite aux lettres à taille limitée n'est pas une solution intéressante. Je me repète: un serveur style OUF mais accessible par le 3615 (éventuellement par le 3614) est certainement la meilleure solution:

- tout message déposé dans la messagerie est visible par tout le monde, exemple: UNTEL1 à SECRETAIRE, UNTEL2 à UNTEL1.

Une seule commande permet d'obtenir l'identification du déposant, du destinataire, le sujet du message, ceci sans confidentialité:

message 355:
  28/01/88 20h45: de SYSOP à TOUS - HORAIRES MAINTENANCE
message 356:
  28/01/88 20h55: de ....

suite page 11

```
[  0] N DEPLACE A;⎕IO
[  1] ⎕IO←1
[  2] A←'ABC'⍳A
[  3] →(0=N)/0
[  4] →(0=∧/A∊ 1 2 3)/ERR
[  5] →(2≠⍴A)/ERR
[  6] (N-1)DEPLACE 'ABC'[(1↑A),6-+/A]
[  7] (3 0 ⍕NC←NC+1),':  DEPLACER LE DISQUE  ',(1 0 ⍕N),'  DE  ',('ABC'[1↑A]),'  EN  ','ABC'[¯1↑A]
[  8] TOURS
[  9] (N-1)DEPLACE 'ABC'[(6-+/A),¯1↑A]
[ 10] →0
[ 11] ERR:'DEPLACE: erreur de données'
```

```
[  0] TOURS;⎕IO;L;LL;M
[  1] 0⍴⎕INKEY
[  2] ⍎'T',(1 0 ⍕1↑A),'←1↓T',(1 0 ⍕1↑A)
[  3] ⍎'T',(1 0 ⍕¯1↑A),'←N,T',(1 0 ⍕¯1↑A)
[  4] M←⍉⊖(3,NO)⍴(NO↑⌽T1),(NO↑⌽T2),NO↑⌽T3
[  5] I←⎕IO←1
[  6] DEB:L←'     ',(9↑⌽LL),LL←(1 0 ⍕M[I;1]),(M[I;1]⍴⎕AV[221]),((9-M[I;1])⍴' ')
[  7] L←L,'     ',(9↑⌽LL),LL←(1 0 ⍕M[I;2]),(M[I;2]⍴⎕AV[221]),((9-M[I;2])⍴' ')
[  8] L←L,'     ',(9↑⌽LL),LL←(1 0 ⍕M[I;3]),(M[I;3]⍴⎕AV[221]),((9-M[I;3])⍴' ')
[  9] L
[ 10] →((1↑⍴M)≥I←I+1)/DEB
[ 11] '     ',63⍴⎕AV[206]
[ 12] (12⍴' '),'A',(21⍴' '),'B',(21⍴' '),'C'
[ 13] ''
```

```
PROBLEME DES TOURS DE HANOI
TROIS TOURS  A  B  et  C
          TOUR DE DEPART: B
          TOUR D'ARRIVEE: C
        NOMBRE DE DISQUES: 3

     A                    B                    C

Presser une touche pour chaque deplacement

1:  DEPLACER LE DISQUE  1  DE  B  EN  C

     A                    B                    C

2:  DEPLACER LE DISQUE  2  DE  B  EN  A

     A                    B                    C

6:  DEPLACER LE DISQUE  2  DE  A  EN  C

     A                    B                    C

7:  DEPLACER LE DISQUE  1  DE  B  EN  C

     A                    B                    C

ET VOILA !
```

Chaque connexion est enregistrée. Si vous avez une BAL (pseudo + code) le système vous signale le nombre de message vous étant destinés depuis la dernière connexion:

dernière connexion SYSOP: 28/01/88 à 20h45
dernier message reçu: 351
nombre de messages reçus: 4
taper L pour les lire, R pour le résumé

Le mode menu permet de sélectionner une option:

TELECHARGEMENT
CHANGEMENT PSEUDO et CODE
ECRIRE AU SYSOP
ECRIRE UN MESSAGE
LIRE UN MESSAGE
SUPPRIMER UN MESSAGE
entre autres commandes.

Les enregistrements n'entrant pas dans le contexte des activités de l'association seraient supprimés (messages roses, ventes illégales de matériel ou de logiciel de provenance douteuse, essais cafouilleux, injures, etc...).

La taille d'un message serait variable. Profitons de ce que le MINITEL puisse exploiter le scrolling (défilement vertical) pour simplifier la manipulation de la messagerie. De plus, le téléchargement de petits programmes en ASCII serait simplifié.

Pourquoi le 3615: transmission en 1200 bauds dans le sens serveur-usager, donc débit correct. OUF ne transmet qu'à 300 bauds en réseau commuté avec terminal et modem classique, à 1200/75 bauds avec un minitel mais sans téléchargement (transmission de fichiers compressés). L'accès T3 3615 est moins coûteux que le réseau commuté et permet l'accès multiple. C'est aussi le seul moyen pour qu'une société accepte de s'occuper de la maintenance

suite page 15

COURRIER:

** HUMEURS ... ***

A propos de fichiers texte... Je relève dans BYTE de Janvier 87 quelques réflexions puisées dans BIX (le réseau BAL des utilisateurs, de Forth en particulier), à propos de la conversion fichiers-blocs:

"Il est facile de transformer des blocs en fichiers ASCII, et vice versa... lisez 64 caractères, ôtez les 'trailing spaces', ajoutez le CR (ou CR/LF) et écrivez le tout dans un fichier texte. Inversement, lisez une ligne de texte, ôtez-lui sa 'fin de ligne', ajoutez des espaces si elle fait moins de 65 caractères et écrivez-la dans un bloc; sinon coupez-la à l'espace précédant le 64ème caractère."

Oui, mais:

"Des écrans en fichiers ASCII, oui. L'inverse, non. Car où s'arrête une définition? Il est certainement possible de convertir 2 secteurs (d'un disque) en écrans, mais ces écrans seront "inchargeables"".

Et j'ajouterais: qu'en est-il des erreurs? A moins que vous soyez de ces programmeurs qui ne font jamais de fautes de frappe ou de syntaxe...

Cependant:

" Il suffit de couper les lignes seulement sur des espaces (i. e. ne pas couper un mot) et la conversion fichier-écran marche très bien."

C'est d'un optimiste.

Mais la question est: pourquoi cette obstination à vouloir écrire un programme avec un autre éditeur que le 'standard', et parfois même avec un traitement de texte (utiliser WordPerfect pour créer un source Forth, c'est, mon cher Secrétaire, prendre CRAY 1 pour tenir votre compte courant !). Je crois que la réponse est donnée par Laxen (ou Perry) lui-même: l'éditeur 'standard' est une "abomination". Vrai. Mais il en existe d'autres, et JEDI en a publié au moins 2. Alors?

Appliquons la méthode expérimentale chère à DESCARTES:

1 - Compilation du programme FBASEII, 25 écrans sur un noyau "vide": 28 secondes;

2 - Conversion en fichier "texte", mise au point (ça ne marche jamais du premier coup) et compilation: 1 minute 32 secondes.

Je croyais que TURBO signifiait, entre autres, 'accélérer la compilation'. Si le temps en est multiplié par 3,3, le terme n'est certainement plus justifié. D'ailleurs, pourquoi se compliquer la vie avec des programmes de conversion? Vous avez un source ASCII en syntaxe Forth, faites simplement sous DOS:

A> F83 < SOURCE.TXT

et ça marche! A condition que votre source soit adapté (pas de LF, et il faut ajouter un SAVE-SYSTEM xxx.COM suivi d'un BYE à la fin du source ...). Mais dans tous ces cas, une grosse partie du problème a été éludée, car on prend un programme déjà mis au point avec l'"abomination" ou son équivalent (ou on construit une boucle de 3 lignes pour voir...). Mais jouez le jeu à partir d'un éditeur extérieur à Forth, en commençant à la première ligne d'un programme en comportant 1200 (taille de FBASEII), et notez combien de fois vous supportez de sortir de l'éditeur pour appeler Forth, compiler, ré-entrer, corriger,...

Reste deux solutions:

1 - créer ou copier (ou pirater) un éditeur intégré plein écran (cf par exemple JEDI n° 37, Juin 87). La structure en blocs des sources Forth n'est pas un dinosaure: celà permet entre autres de tester une petite fraction d'un programme.

2 - si vous tenez absolument à utiliser un éditeur ou un traitement de texte 'extérieur', appelez-le à partir de Forth : la fonction 4B de l'interruption 21 est faite pour. Mais ceci est un autre article... que notre Secrétaire, s'il est bien gentil, publiera dans ce même numéro ... (2 écrans suffisent).

A. Jaccomard, Jan 88
29190 PLEYBEN

LISTING:

```
' ------------------------------------------------------------ ' ' ---------------------------------------------------------- '
'   Ce programme donne accès au fichier COMMAND.COM de MS-DOS. ' '  Utilisation :  COM <Enter>                               '
'   L'accès n'est pas limité aux seules commandes du DOS: tout ' '     le système demande un nom de commande.                '
' programme peut être lancé, pourvu qu'il tienne  dans la RAM  ' ' - si vous entrez une commande DOS (ou un nom "exécutable"), '
' disponible, la partie de la mémoire occupée  par  F83  étant ' '  elle est exécutée, et le retour sous Forth est automatique.'
' "réservée" (par un appel à la fonction 4AH de l'INT21H).     ' ' - si vous frappez <Enter> seul, vous êtes sous DOS, comme  '
'   On peut aussi créer des "mots" d'appel spécialisés.        ' '  indiqué par son "prompt", utilisation tout à fait normale. '
'   Par exemple, si votre éditeur préféré est MONEDIT.EXE,     ' '  Le retour sous Forth se fera par :                        '
' remplacez COMMAND.COM dans COM par MONEDIT.EXE et changez    ' '        A>EXIT <Enter>                                     '
' COM en EDITEUR.                                              ' '                                                           '
' ------------------------------------------------------------ ' ' ------------------------------------ A. Jaccomard, Jan 88 '
```

```
     2
\ Appel à COMMAND.COM                                   12Jan88JaD
CREATE BUF-CMD 10 ALLOT          \ BUF-CMD sera pointée par DS:DI
128 CONSTANT COMMANDE            \ tampon commande DOS (et DTA)
: CMD?   (S -- )                 \ prend la commande ...
  CR ." Quelle commande?"  COMMANDE DUP 64 ERASE
  CR  DUP 4 + 80 EXPECT  SPAN @  ?DUP
     IF                          \ quelque chose est entrée ...
 3 + OVER C!                     \ lgr totale entrée + 3 octets
 1+ 47 OVER C!                   \ syntaxe DOS
 1+ 67 OVER C! 1+ 32 OVER C!     \ "/C  "
 SPAN @ + THEN 1+ 13 SWAP C! ;   \ ... sinon juste un CR.
CODE CS?  CS PUSH NEXT C;        \ Code Segment empilé.
CREATE LIST-PARAM                \ sera pointée par ES:BI
  0 , ( environmt) COMMANDE , CS? ,   ( adr ligne code)
  92 , CS? , ( FCB1 en 005CH ) 108 , CS? , ( FCB2 en 006CH )
                                                        -->
```

```
     3
\ Appel à COMMAND.COM                                   12Jan88JaD
  HEX
CODE  DOS-COMMAND  (S -- )
  1000 # BX MOV  4A # AH MOV  21 INT   \ réserve 64K
   BUF-CMD # DX MOV                    \ DS:DX pointe 'Cmd'
   LIST-PARAM # BX MOV                 \ et ES:BX sur 'param'
  BP PUSH  DS PUSH  ES PUSH            \ sauve le contexte
  SP@ # AX MOV AX FF04 #) MOV          \ le pointeur de pile,
  SP0 #) AX MOV  AX FF06 #) MOV        \ et la pile elle-même
  4B00 # AX MOV  21 INT                \ appel au DOS
  FF04 #) AX MOV AX SP0 # MOV  FF06 #) AX MOV AX SP0 #) MOV
  ES POP  DS POP  BP POP               \ restitue pile et contexte
  NEXT C;  DECIMAL
: COM  (S -- )                         \ mot utilisateur
  " \COMMAND.COM"  BUF-CMD DUP 18 ERASE SWAP CMOVE
   CMD? DOS-COMMAND ;
```

Objectif Division et racine carree de grands nombres. Curiosités mathématiques et antiquités.
Langage FORTH bien évidemment
Difficulté y'en a pas ou si peu
Matériel ça tourne sur ATMOS mais ça devrait tourner sans problème sur du petit matériel comme les compatibles PC
But Me faire plaisir (c'est déjà pas mal !) et à d'autres par la même occasion.
Date 25/12/87 si ! si ! C'est bien le jour où on se fait plaisir.

Je ne vais pas cracher dans la soupe: le FORTH c'est bien mais le problème est que lorsqu'on veut faire des opérations il faut toujours savoir la grandeur des nombres; si le forth va les comprendre comme signés ou non, etc..
A la longue, ça devient gênant surtout si on ne peut pas le prévoir à l'avance.
J'étais en train de faire un programme de géométrie (et bien oui ! encore des maths) assistée par ordinateur: dessin et calcul automatique d'intersections, de médiatrices,de symétries,de projections etc... quand je me suis trouvé coincé par le manque de précision des calculs. Comment faire ? Changer de méthode ? Non ! déjà que faire des calculs de pente qu'avec des nombres entiers,ce n'est pas de la tarte !
Solution: lui apprendre à faire des divisions !

I Divisions de grands nombres positifs
Le problème est de faire des divisions qu'avec des nombres positifs et qu'ils soient bien compris comme tels.
Faire 234 2 / pas de problème 23456 2 / non plus
64000 2 / ah déja un problème car 64000 est compris comme négatif sur 16 bits
solution le passer en double précision sur 32 bits et employer MU/MOD ou UM/MOD
donc faire 64000. 2 MU/MOD DROP NIP si on ne veut que le quotient entier 16 bits
de là pour 123456. idem car c'est déjà un nombre 32 bits
Alors où est le problème, aucun si c'est le premier nombre qui augmente,tout est possible !
Mais si le deuxième devient un 16-bits signé négatif et à forciori un 32-bits.
Enfin bizarre ! car pour un diviseur comme 33000 qui devrait être un 16-bits signé MU/MOD et UM/MOD donnent chez moi le même résultat pour par ex: 123456. 33000 MU/MOD donne 3
mais 123456. 64000 MU/MOD donne pour quotient 0 et reste: -7616 (et UM/MOD pareil)
Est-ce que ça dépend de l'implantation ! (Ne t'inquiètes pas Michel) mais quelqu'un pourrait-il m'expliquer l'algorithme de division en LM de UM/MOD . Oh si dites !!! Comme étrenne !
Toujours est-il qu'il y a de quoi perdre son latin ou plutôt ses tables de multiplication.
Ma solution: refaire les divisions comme on les ferait à la main: vous savez: *En tout ça,. combien de fois l'autre; j'essaie tant,tant fois ça,... ôté de ... il reste ... et je retiens ... etc... Ah zut c'est trop grand... je le diminue et on recommence.Puis on abaisse...* Et ça marche, sauf que l'on est pas du tout obligé de le faire en base 10 On le fait directement en base 256, la méthode reste valable. La différence est que le plus grand chiffre,si on peut appeler ça un chiffre est 255.
Si on travaillait en assembleur,on aurait interêt à faire la même chose en base 2.
Les mots importants:
ENTD qui permet d'entrer le dividende 32-bits. ex: 1234567890. ENTD
ENTS qui entre le diviseur 32-bits de la même façon. ex 9876543. ENTS
Ces deux mots utilise le même ENTRE qui met dans les zones DIVD et DIVS les nombres 32-bits en base 256 partie haute en avant comme on lit naturellement un nombre. Pourquoi ? car selon cette méthode, pas besoin d'une routine pour comparer deux nombres, le mot COMP fait l'affaire.
DIVISE vous vous en doutez met le quotient entier dans la zone QUOT . Le reste est alors dans la zone DIV2 qui contient les dividendes utiles successifs comme à la main.
Ce mot ne laisse rien sur la pile (car je l'utilise par la suite pour autre chose)
Les mots QUOT FREND met le quotient 32-bits sur la pile.
DIV2 FREND met le reste sur la pile.
.QUOT et .RESTE l'afficherait directement.
Au passage on peut noter /QR qui laisse le quotient et le reste dans la base courante.
R*/ fait une multiplication avec retenue.
DECALG qui fait un décalage vers la gauche de la zone considérée en faisant rentrer à droite un octet,et sortir à gauche un octet sur la pile. Ca ressemble à de l'assembleur.
Les 3 premiers écrans suffisent.
Mais puisque cette méthode ést générale,elle marche pour des nombres beaucoup plus grands.
Ca devient purement théorique car ces nombres ne peuvent pas être maniés par le forth sur la pile par exemple.
Il suffit de travailler sur leur écriture décimale: le nombre 124567895 est découpé en 1 24 56 78 95 .
Bien sur on travaille en base 100,ca marche tout aussi bien. Seules les opération d'entrées-sorties changent.
DIVIDENDE permet de rentrer le premier nombre compris comme caractère décimal. ex: DIVIDENDE 11324576890I
DIVISEUR fait de même avec le diviseur. La division se fait par DIVISE.
Puisque le résultat n'est pas un nombre au sens propre,mais seulement une écriture;on ne peut qu'afficher les résultats par .QUOT et .RESTE
Un seul écran suffit pour cela: le 4
Pour cela,il faut que les zones qui vont contenir les nombres soient assez grandes. Si leur longueur de 4 suffit pour la première version 32-bits,leur longueur doit être élastique pour l'autre. Elle est fixée par la constante LG mise à 10 à l'initialisation.Mais il est bien évident que les 6 dernières ne servent à rien et même font perdre du temps pour la première façon.
Donc avec des zones de 10,on peut faire des divisions de 20 chiffres décimaux.
note Dans mes tiroirs,j'ai un autre programme de division (juste 13 écrans) qui permet de réapprendre à faire des divisions (avec virgule). On récite la litanie habituelle (voir au dessus) avec démonstration et exercices.
C'est un péda(nt?)gogiciel quoi ? Si ça interesse quelqu'un !
Et ça marche bien sûr dans n'importe quelle base,puisque c'est en forth.

II Racine carrée de nombres (petits ou grands)((de 7 à 77 ans))

Cette question commence à revenir régulièrement dans JEDI.
Mais il n'y avait aucune solution qui vraiment me plaisait,comme sûrement celles-ci ne vous plairont pas entièrement.
La première donnée dans le numéro 27 est basée (si j'ai tout compris) sur la méthode de NEWTON.

où $X_{N+1}=(X_N+A/X_N)/2$ A étant le nombre dont on cherche la racine (carrée ici)
Méthode par récurrence où quand N tend vers l'infini, X_N tend vers cette racine. Ce qui est interessant c'est que l'infini n'est pas loin ! (converge rapidement)
Ce qui me gênait c'était la boucle de 10 0 DO... Je l'ai modifiée en un BEGIN...WHILE...REPEAT et on arrête quand les calculs ne donnent plus de différence.

La seconde donnée par le fig-Hambourg était basée sur la somme des premiers nombres impairs qui est carrée (si j'ai bien compris la aussi.) C'est bien mais la encore,si les nombres sont des 16-bits signés,que se passe-t'il ?

Ma solution est basée sur NEWTON.

SQRT donne les racines carrées de nombres jusqu'à 32767 (16-bits signés positifs)
USQRT donne en plus les racines jusqu'à 65535.
Si vous regardez bien,il a fallu passer par des divisions en 32-bits.
DSQRT enfin me plait pas mal,mais elle ne donne que les racines carrées inférieures à 32767 donc pour des nombres 32-bits inférieurs à 32767x32767=1073676289.

Aucun de ces mots ne teste le fait de savoir si jamais le nombre ést négatif. L'avait qu'à faire gaffe ! non mais !
L'un des problèmes est de savoir par quel nombre x_0 commencer;car plus on prend un nombre près de la racine, plus on est vite arrivé. Evident non ! J'ai choisi arbitrairement la moitié du nombre de départ. En fait ça ne fait gagner qu'une division;et 32767 pour les 32-bits car évidemment la racine ne peut être plus grande.
En général,c'est assez rapide et on comprend le pourquoi de la boucle 10 0 DO... car 10 calculs suffisent bien dans les cas 16-bits. Pour les nombres 32-bits purs,il faut plus de calculs de 15 à... 2. Une quinzaine pour de nombres comme 68000. et 3 pour 1073676289. Evident car pour le premier on est très loin du premier nombre essayé:32767

Tout ça tient en 1 écran le cinquième.A la fin de cet écran, il y a un second DSQRT qui donnerait des racines plus grandes si les divisions forth tenaient la route.

Et justement puisque maintenant on sait faire de grandes divisions, on peut chercher de grandes racines carrées.
C'est le but de l'écran 6 où MSQRT donne les racines de très grands nombres 32-bits s'ils sont rentrés avec ENTD ou décimaux s'ils sont rentrés par DIVIDENDE.
ex: faire 4194240000. ENTD MSQRT .QUOT ou DIVIDENDE 12345678909987651 MSQRT .QUOT

Problème ce n'est pas rapide,mais alors pas du tout (10 à 20 s pour des nombres corrects)
Comment aller plus vite. Première solution employée: si le nombre est 32-bits,la racine est inférieure à 65535.Evident
Pour les autres,on se base sur l'écriture,la racine d'un nombre ne peut pas prendre plus de la moitié des chiffres du nombre de départ à 1 chiffre près. Pour un nombre de 15 chiffres, la racine n'aura que 8 chiffres maximum. on commence alors au plus grand nombres de 8 chiffres qui est 99999999 .

Alors ça marche ? Oui mais c'est encore lent car on répète plusieurs fois des divisions qui déjà prennent du temps.

Comment calculer la racine d'un grand nombre avec un ordinateur:La calculer à la main

Non ! Ne balancez pas votre petit PC (ou pas trop loin de chez moi que je...),on va faire comme à la main.
Vous savez:la méthode qui ressemble à une division,qui a le goût d'une division (donc on aime pas) mais qui n'est pas une division. On l'apprenait autrefois à l'école !
En voici un exemple: soit à calculer la racine de 119737 .
On découpe ce nombre en tranches pas trop épaisses (C'est bien une recette) de 2 chiffres par la droite.

```
11 97 37 !3/34/346
-9       !-----
--       !60x0=0 ;61x1=61;62x2=124
 2 97    !63x3=189; 64x4=256
-2 56    !-----
 ----    !680x0=0;681x1=681
   41 37 !682x2=1364 ....
  -41 16 ! 686x6=4116
  ----- !
      21 !
         !
```

- On prend la première tranche 11 . Le plus grand carré inférieur est 9 et c'est le carré de 3. Le premier chiffre est donc 3
- On abaisse la deuxième tranche 97 et là commence le cirque.
  On double les chiffres trouvés.3x2=6
  On cherche le plus grand nombre 6NxN qui puisse être enlevé à 297
  Le plus grand possible est 256 donc le chiffre suivant est 4;
  65x5 était trop grand.On soustrait 256 et on repart du reste 41
- On abaisse la tranche 37. On double les chiffres trouvés:34 -> 68
  La plus grande posibilité est obtenue avec 686x6=4116
  donc le troisième chiffre est 6
- Si on voulait continuer après la virgule, on abaisserait la tranche 00.
  l'interêt est aussi que l'on connait le reste de cette opération.
  Le problème est qu'il faut essayer tous les chiffres successivement à partir de 0.

Encore une fois,la méthode est valable pour toutes les bases. Le tout est de l'adapter.
Quelle base employer:Dans la division sur l'écriture décimale,on employait la base 100;mais ici une tranche de 2 chiffres ne donne qu'un seul chiffre au 'quotient'. Donc des octets de 1 à 256 donneront des chiffres de 1 à 15 ; on va travailler en hexadécimal.On a l'habitude avec le forth.
Rien à dire,les mots de l'écran 8 suivent cette démarche.C'est plus rapide que MSQRT mais c'est quand même lent.
Il faut faire 1234567. ENTD puis RACIN1 puis .QUOT ou .RESTE
ou bien DIVIDENDE 1234567 RACINE1 etc...
note: pour le mot REQUOT il faut que LG soit pair.Les chiffres arrivent 1 par 1,et pour les remettre dans le format d'écriture,on les écrit 2 par 2 ex: 3 4 6 va donner 3 46 en décimal.

Est-ce possible d'être plus rapide Réponse Oui ! (sinon je n'aurais pas posé la question !
Comment:en modifiant ce qui est le plus long.La recherche du chiffre par des multiplications successives.
Si on regarde les calculs effectués et les différences entre les produits successifs:
entre 680x0 et 681x1 différence de 681 entre 681x1 et 682x2 différence de 683 entre 682x2 et 683x3 différence de 685 entre 683x3 et 684x4 différence de 687 etc... vu ?
les différences sont 681,683,685,687 donc de 2 en 2 à partir de 681 (non 0)
Au lieu de les soustraire qu'à la fin,on les soustrait au fur et à mesure lorsque c'est possible (d'où la ressemblance avec une division;la différence étant que les nombres soustraits ne sont pas les mêmes.)
La dernière méthode employée est celle-là: écran 7 et elle est deux fois plus rapide que l'autre.

Est-ce possible d'être plus rapide Réponse :Je n'en sais rien ! Oui si on passe à l'assembleur.(mais l'appliquant au binaire et ça devient plus simple car: soit le reste partiel est plus petit alors chiffre 0
soit il est plus grand et le chiffre ne peut être que 1)

Mais en améliorant la méthode et en restant en forth. ?
Je vous écoute !!

Amicalement Jean-luc SIRET

```
FILE: BIGDIV.FTH / SCRN# 1
\ Division double:preliminaires
ONLY FORTH DEFINITIONS DECIMAL
VARIABLE ESSD VARIABLE RET VARIABLE OCT VARIABLE FL
VARIABLE OCT1          10 CONSTANT LG  ( pair )
: :TAB CREATE LG 1+ ALLOT DOES> ;
:TAB DIVD :TAB DIVS :TAB RESP :TAB QUOT :TAB DIV2
: /QR 0 OCT @ MU/MOD DROP ;
: R*/ *D RET @ 0 D+ OCT @ MU/MOD DROP ;
: CHO 1 BEGIN 2DUP + C@ 0= WHILE 1+ REPEAT SWAP C! ;
: ENTRE 256 OCT ! >R /QR R LG 3 - + C! R LG 2- + C!
        /QR R LG 1- + C! R LG + C! R> CHO ;
: VIDE LG 1+ ERASE ;
: ENTD DIVD DUP VIDE ENTRE ;
: ENTS DIVS DUP VIDE ENTRE ;
: DECALG DUP 1+ C@ -ROT DUP 2+ DUP 1- LG 1- CMOVE LG + C! ;
-->
```

```
FILE: BIGDIV.FTH / SCRN# 2
\ Division double
: MULT1 0 LG DO DIVS I + C@ ESSD @ R*/ RET !
              RESP I + C! -1 +LOOP ;
: MULT 0 RET ! MULT1 ;
: SOUS 0 RET ! 0 LG DO DIV2 I + C@ RESP I + C@ RET @ + -
      DUP 0< IF OCT @ + 1 RET ! ELSE 0 RET !
      THEN DIV2 I + C! -1 +LOOP ;
: DIV1 ESSD @ 0 MAX OCT @ 1- MIN ESSD !
      BEGIN MULT RET @ 0= NOT
      DIV2 1+ RESP 1+ LG COMP -1 = OR WHILE
      -1 ESSD +! REPEAT SOUS ;

: PRD1 DUP C@ OCT @ *D ROT 1+ C@ 0 D+ DROP ;
: PREND DUP LG 1- + PRD1 SWAP LG 3 - + PRD1 ;

-->
```

```
FILE: BIGDIV.FTH / SCRN# 3
\ Division double
: ?QUOT DIV2 DUP C@ + C@  DIVS DUP C@ + C@ DIV2 C@ DIVS C@
        < IF >R OCT @ *D DIV2 DUP C@ 1+ + C@ 0 D+
             R> MU/MOD DROP NIP ELSE / THEN ESSD ! ;
: DIVISE RESP VIDE QUOT VIDE DIV2 VIDE
        LG 0 DO 0 DIVD DECALG DIV2 DECALG DROP DIV2 CHO
        DIV2 1+ DIVS 1+ LG COMP -1 =
        IF 0 ESSD ! ELSE ?QUOT DIV1 THEN
        ESSD @ QUOT DECALG DROP LOOP ;
\ : .RESTE DIV2 PREND D. ;
\ : .QUOT  QUOT PREND D. ;




-->
```

```
FILE: BIGDIV.FTH / SCRN# 4
\ Division decimale
: ENT2 DUP VIDE BL WORD DUP C@ DUP >R 2 MOD
     1 = IF 1+ C@ 48 - OVER DECALG DROP HERE 1+ THEN R>
     2/ 0 ?DO DUP I 2* 1+ + DUP C@ 48 - 10 * SWAP 1+ C@ 48 - +
     >R OVER R> SWAP DECALG DROP LOOP DROP CHO 100 OCT ! ;
: DIVIDENDE DIVD ENT2 ;
: DIVISEUR  DIVS ENT2 ;
: AFFOCT 48 + DUP 48 = IF DROP RET @ IF 48 EMIT THEN
                       ELSE EMIT RET ON THEN ;
: AFFICHE SPACE RET OFF LG 1+ 1 DO DUP I + C@ 10 /MOD AFFOCT
          AFFOCT LOOP DROP RET @ 0= IF 48 EMIT THEN SPACE ;
: AFF2 OCT @ 256 = IF PREND D. ELSE AFFICHE THEN ;
: .RESTE DIV2 AFF2 ;
: .QUOT  QUOT AFF2 ;

-->
```

```
FILE: BIGDIV.FTH / SCRN# 5
\ Racine carree , simple ,simple non signee, double
ONLY FORTH DEFINITIONS DECIMAL
: U/MOD 0 SWAP MU/MOD DROP ; : U/ U/MOD NIP ;

: SQRT  DUP 1+ 2/ BEGIN 2DUP / OVER + 2/
        2DUP - 0 > WHILE NIP REPEAT MIN NIP ;
: USQRT DUP 1+ U2/ BEGIN 2DUP U/ OVER + U2/
        2DUP - 0 > WHILE NIP REPEAT MIN NIP ;
: DSQRT DUP 0= IF OVER U2/ 1+ ELSE 32767 THEN
        BEGIN 3DUP MU/MOD DROP NIP OVER + U2/
        2DUP - 0 > WHILE NIP REPEAT MIN NIP NIP ;
\ : DSQRT 65535 BEGIN 3DUP MU/MOD DROP NIP
\          OVER 0 SWAP 0 D+ 2 MU/MOD DROP NIP
\          2DUP 0 SWAP 0 DNEGATE D+ 0< IF DROP -1 THEN
\          0 U> WHILE NIP DUP U. REPEAT MIN NIP NIP ;
-->
```

```
FILE: BIGDIV.FTH / SCRN# 6
\ Racine carree par NEWTON
: 2DIV  DIVS VIDE 2 DIVS DECALG DROP DIVS CHO ;
: ADD DIVD VIDE 0 RET ! 0 LG DO DIVS I + C@ QUOT I + C@ +
          RET @ + /QR RET ! DIVD I + C! -1 +LOOP ;
: MSQRT DIVD PAD LG 1+ CMOVE OCT @ 256 = IF 65535. ENTS
        ELSE DIVS VIDE LG DIVD @ - 1+ 2 /MOD +
             0 DO 99 DIVS DECALG DROP LOOP
        THEN DIVS PAD LG 1+ + LG 1+ CMOVE
        BEGIN PAD          DIVD LG 1+ CMOVE
              PAD LG 1+ + DIVS LG 1+ CMOVE DIVISE
              ADD DIVD CHO 2DIV DIVISE QUOT CHO
              QUOT 1+ PAD LG 2+ + LG COMP -1 = WHILE
              QUOT PAD LG 1+ + LG 1+ CMOVE REPEAT
              PAD LG 1+ + QUOT LG 1+ CMOVE ;
\ faire ENTD ou DIVIDENDE puis MSQRT et .QUOT
-->
```

```
FILE: BIGDIV.FTH / SCRN# 7
\ Racine carree, ancienne methode LG PAIR !!
: REQUOT DIVS VIDE 0 LG DO QUOT I + DUP C@ SWAP 1- C@ OCT1 @ *
         + LG DUP I - 2/ - DIVS + C! -2 +LOOP ;
: 2ADD 2 RET ! 0 LG DO RESP I + C@ RET @ + /QR RET ! RESP I +
       C! RET @ 0= ?LEAVE -1 +LOOP ;
: 2OA+1 OCT1 @ 2* ESSD ! MULT 1 RESP LG + +! 1 ESSD ! ;
: RAC1 REQUOT RESP VIDE 2OA+1 BEGIN
       DIV2 1+ RESP 1+ LG COMP -1 = NOT ESSD @ OCT1 @ = NOT AND
       WHILE SOUS 2ADD 1 ESSD +! REPEAT -1 ESSD +! ;
: RACINE DIV2 VIDE QUOT VIDE FL OFF
         OCT @ 256 = IF 16 ELSE 10 THEN OCT1 !
         LG 0 DO 0 DIVD DECALG DUP DIV2 DECALG DROP
         0= FL @ 0= AND IF 0 ESSD ! ELSE RAC1 FL ON THEN
         ESSD @ QUOT DECALG DROP LOOP
         REQUOT DIVS QUOT LG 1+ CMOVE ;

```

```
FILE: BIGDIV.FTH / SCRN# 8
\ Racine carree, ancienne methode LG PAIR !!
\ : REQUOT DIVS VIDE 0 LG DO QUOT I + DUP C@ SWAP 1- C@ OCT1 @
\          * + LG DUP I - 2/ - DIVS + C! -2 +LOOP ;
\ : 2OXA OCT1 @ 2* ESSD ! MULT RESP DIVS LG 1+ CMOVE 0 ESSD ! ;
\ : MULT2 RESP VIDE ESSD @ DUP * RET ! MULT1 ;
\ : RAC1 REQUOT RESP VIDE 2OXA BEGIN 1 ESSD +! MULT2
\       DIV2 1+ RESP 1+ LG COMP -1 = ESSD @ OCT1 @ = OR UNTIL
\       -1 ESSD +! ESSD @ 0<> IF MULT2 SOUS THEN ;
\ : RACIN1 DIV2 VIDE QUOT VIDE FL OFF
\          OCT @ 256 = IF 16 ELSE 10 THEN OCT1 !
\          LG 0 DO 0 DIVD DECALG DUP DIV2 DECALG DROP
\          0= FL @ 0= AND IF 0 ESSD ! ELSE RAC1 FL ON THEN
\          ESSD @ QUOT DECALG DROP LOOP
\          REQUOT DIVS QUOT LG 1+ CMOVE ;
```

logicielle et matérielle sans contrepartie financière autre que les bénéfices provenant des connexions.

Une autre solution: CALVACOM en 3613 avec cité JEDI: 60 Fr de l'heure par abonnement, donc au même prix que le 3615. Un serveur 3615 code JEDI donnerait une vraie vitrine à l'association et serait le moyen le plus efficace pour communiquer rapidement des questions, des réponses et de petits trucs et astuces. De son coté JEDI se proposerait de reprendre dans ses colonnes tous les messages ayant un intérêt collectif et les réponses afférentes à ces messages.

```
------------------------------------------------------------
:                                                          :
:          GESTION DE FICHIER SOUS TURBO-PASCAL            :
:                                                          :
------------------------------------------------------------
```

par Eric RAMAHEFARIVONY

Turbo-PASCAL

Dans l'esprit de la version 4.0 qui permet d'accéder à toutes les fonctions du système d'exploitation sans quitter le TURBO, cet utilitaire offre les fonctions de base de gestion de fichier (efface, renomme, copie). Il a été développé sur AMSTRAD sous CP/M 2.2 avec TURBO 3.0 et a été paramétré (taille du buffer de transfert) pour que le source et l'objet compilé tiennent simultanément en mémoire.

Les options par défaut sont entre crochets. Ainsi, pour sortir d'un menu, on tapera RETURN. Le suffixe '.pas' est optionnel. Le symbole '*' est subtitué par la racine d'un nom de fichier saisie au préalable. C'est une variable globale dont la valeur est conservée entre deux appels de l'exécutable autorisant l'appel des utilitaires internes au TURBO (tels que le (D)ir, ou (L)e changement de disquette).

Liste du programme io.pas :

```
program   io;
type      chaine = string [14];
const     CAN : char = #24; (* reverse *)
          nom : chaine = '';
          racineCourante :  boolean = FALSE;
var       filVar : file;

function  saisie( default : char) : char;
var       reponse : char;
begin  (* saisie lit un caractere au vol *)
repeat until keyPressed;
read( kbd, reponse); reponse := upCase(reponse);
if reponse = chr(13) then reponse := default;
writeln( reponse );
saisie := reponse
end;   (* saisie *)

procedure saisieNom( msg : chaine; VAR nomFile : chaine);
var       star : byte; nomLocal, racine : chaine;
begin  (* saisieNom saisie un nom de fichier *)
if not racineCourante then
   racine := copy( {in} nom, 1, pos( '.', nom) - 1);
write( CAN, msg, CAN, ' : (*="', racine, '") [CR] ? ');
nomLocal := ''; readln(nomLocal);
if nomLocal <> '' then
   begin
   if pos( {in} '.', nomLocal) = 0 then
      nomLocal := nomLocal + '.pas';
   star := pos( {in} '*', nomLocal);
   racineCourante := star > 0;
   if racineCourante then
      begin
      delete( {in} nomLocal, {from} star, {nbr} 1);
      insert( {what} racine, {in} nomLocal, {from} star);
      end;
   end;
nomFile := nomLocal
end;   (* saisieNom *)

function  existeFile(fileName: chaine; msg: char):boolean;
var       filExist : file; existe : boolean;
begin  (* existeFile teste l'existence d'un fichier *)
assign( {out} filExist, {in} fileName);
{$I-} reset( filExist ); {$I+}
existe := IOresult = 0;        (* ouverture en lecture ok *)
if existe then close( filExist );
case msg of
     '+' : if     existe then
                  writeln( fileName,' existe DEJA');
     '-' : if not existe then
                  writeln( fileName,' N''existe PAS')
     end;
existeFile := existe
end;   (* existeFile *)

procedure filout( enableDestroy:char; VAR newName:chaine);
var       existe : boolean;
begin  (* filout saisie le nom du fichier destination *)
repeat saisieNom( éiné 'fichier dest.', éouté newName);
       if (newName = '') or (enableDestroy = 'K')
          then existe := FALSE
          else existe := existeFile( newName, '+');
       until not existe
end;    (* filout *)

procedure efface;
begin  (* efface un fichier avec confirmation *)
write('confirme l''effacement de ',nom,' (o/[n]) ? ');
if saisie( 'N' ) = 'O' then
   begin erase( filVar ); writeln( nom,' est efface')
   end
end;    (* efface *)

procedure transfert( nameCible : chaine);
const     secteurSize = 128; bufSize = 8;
var       cible : file; nbrRecord : integer;
          buffer: array[1..secteurSize,1..bufSize] of byte;
begin  (* transfert copie un fichier dans un autre *)
assign( {out} cible, {in} nameCible);
reset( filVar ); rewrite( cible );
repeat blockRead( {in} filVar, {out} buffer,
                  {in} bufSize, {out} nbrRecord);
       if nbrRecord > 0 then
          blockWrite({in} cible, buffer, nbrRecord)
       until nbrRecord = 0;
close( filVar ); close( cible );
writeln(nom,' est copie dans ',nameCible)
end;    (* transfert *)

procedure traite;
var       newName : chaine; reponse : char;
begin  (* traite saisie et lance la commande *)
repeat write('(E)fface (R)enomme (C)opie ');
       write('(K)opieEcrasante [F]in ? ');
       reponse := saisie( 'F' );
       until reponse in ['E','R','C','K','F'];
if reponse <> 'F' then assign( {out} filVar, {in} nom);
case reponse of
     'E': efface;
     'R': begin filout( {in} reponse, {out} newName);
          if newName <> '' then
             begin
             {$I-} rename( filVar, newName); {$I+}
             if IOresult = 0
                then writeln( nom,' <- ', newName)
                else writeln('erreur ',IOresult)
             end
          end;
  'C','K': begin filout( {in} reponse, {out} newName);
          if newName <> '' then transfert( {in} newName)
          end;
     end
end;    (* traite *)


BEGIN  (* principal traite les commandes jusqu'a CR *)
repeat saisieNom( {in} '--> nom[.PAS]', {out} nom );
       if nom <> '' then
          if existeFile( nom, '-') then traite
       until nom = ''
end.    (* principal *)
```

Exemple d'exécution :

on veut sauvegarder le fichier m:io.pas sur la disquette a: sachant que le disque de travail est le virtuel m:

```
Running
--> nom[.PAS] : (*="") [CR] ? io [RETURN]
(E)fface (R)enomme (C)opie (K)opieEcrasante [F]in ? K
fichier dest. : (*="io") [CR] ? a:* [RETURN]
io.pas est copié dans a:io.pas
--> nom[.PAS] : (*="io") [CR] ? [RETURN]

>
```

TRAITEMENTS DE CHAINES

par Charles MOHR

pour F83 Laxen et Perry MSDOS
F83 Laxen et Perry CP/M
TURBO-Forth 83-Standard

Cher secrétaire

Je vous adresse ci-joint un "listing" et une disquette 360k contenant quelques utilitaires de manipulation de chaînes de caractères. Cette version qui est, je l'espère, assez complète, est constituée par la synthèse de mon expérience personnelle en ce domaine, ainsi que des mots glanés au hasard de mes lectures (JEDI...) et adaptés.

J'espère qu'on me pardonnera le coté amateur et les maladresses de ce programme et que les débutants, dont je suis, y trouveront une source d'idées.

A ce propos, en faisant des essais d'application de mon programme, il m'est arrivé plusieurs fois de me "planter" en faisant HERE X ERASE, pour X trop grand. Comment faut-il agir pour pouvoir utiliser la pleine capacité RAM d'un compatible de 640ko, c'est à dire au-delà de LIMIT?

Par ailleurs je recherche un programme de calcul en virgule flottante avec au moins 9 chiffres caractéristiques, tests de dépassement, notation ingénieur et fonctions scientifiques assez complètes (FORTH ou assembleur 8088/86). Si un de vos adhérents possède ça dans ses cartons... A moins que dans le TURBO-F83...?

LISTING

```
\ VCHAIN  (CAPA)  LIMIT)
ONLY FORTH ALSO DEFINITIONS DECIMAL
: VCHAIN (S n <VCHAIN> <nom>  ... )
         ( à l'execution <nom> ... adr ln )
  DUP 1 255 BETWEEN  NOT
  IF . ABORT" Valeur non admise " THEN
  CREATE DUP 0 C, , 4 + ALLOT
  DOES> 2+ COUNT ;
: (CAPA)   ( ...  d )
  'TIB @ S>D PAD S>D 300. D+ D- D2/ ;
: LIMIT)  (S d ... flag v/f )
  (CAPA) D> ;
```

Commentaires de VCHAIN
Utilisé sous la forme: n VCHAIN <nom-de-la-chaine> pour créer une variable chaîne de caractères de n octets max.

Pour la création n est limité arbitrairement à 255 caractères et augmenté de 4 octets "de sécurité". A l'exécution, <nom-de-la-chaine> laisse sur la pile l'adresse explicite et longueur actuelle (réelle).

Par ailleurs :
ADR -1 = adresse implicite utilisée par COUNT
ADR -2 contient n (maximum octets possible)
ADR -3 contient 0 comme repère de chaine
(CAPA) calcule l'espace max alloué aux chaines crées par DIM$
LIMIT) effectue un test de sécurité pour éviter de "planter" le système... Qui me donnera l'astuce permettant d'utiliser les adresses supérieures à LIMIT?

```
\   DIM$
: DIM$ (S n de ch,ln max  <DIM$> <nom> ... )
   ( à l'execution indice <nom> ... adr ln )
2DUP 1 255 BETWEEN NOT SWAP 1 500 BETWEEN NOT OR
  IF SWAP . . ABORT" Valeurs non admises " THEN
  2DUP 4 +  *D  LIMIT)
  IF ABORT" Dépassement limite imposée "   THEN
  CREATE
  2DUP , , 4 + * DUP HERE SWAP ERASE 1+ ALLOT
  DOES> 2DUP 2+ @  U<
  IF      DUP @ DUP >R 4 + ROT * SWAP 4 + +
          DUP 0 SWAP C! DUP 1+ R> SWAP C!
          2+ COUNT
  ELSE    SWAP ." Erreur d'indice pour "
          . 2- >NAME .ID QUIT
  THEN ;
```

Commentaire de DIM$
Utilisé sous la forme n ln DIM$ < nom > pour créer un tableau de chaines de caractères à une dimension:
n = nombre de chaines (indice..) limité arbitrairement à 500.
ln = longueur maximale de chaque chaîne, limitée à 255 caract.

A l'exécution i(ndice) (nom) laissent l'adresse explicite et la longueur réelle de la chaîne. Attention indice max=n-1. Par ailleurs pour chaque chaine:

ADR-1 = adresse implicite utilisée par COUNT
ADR-2 = contient la longueur maximale admise
ADR-3 = contient 0 comme repère de chaine

```
\ (DEBUT?) (2DEBUT?) (LON) LON$
\ MAXLON$ LON$! LON$+!
: (DEBUT?)   ( adr ln ... adr ln si vrai )
  OVER 3 - C@ 0<>
  IF SWAP . . ABORT" N'est pas VCHAIN " THEN ;
: (2DEBUT?) (S adr1 n1 adr2 n2   idem si vrai )
  >R >R (DEBUT?) R> R> (DEBUT?) ;
: (LON) 1- C@ ; ( adr ... lon )
: LON$ (DEBUT?) DROP (LON) ;  ( adr ln ----ln )
: MAXLON$ ( adr ln --- maxln )
 (DEBUT?) DROP 2- C@ ;
: LON$! (S adr ln n ... )
  0 MAX >R OVER SWAP MAXLON$ R>
  MIN SWAP 1- C! ;
: LON$+! (S adr ln n ... )
  >R 2DUP MAXLON$ SWAP R> + 0 MAX DUP ROT >
  IF   DROP 1- COUNT ." Débordement MAXLON$ "
  SWAP . . abort THEN SWAP 1- C! ;
```

Commentaires:
(DEBUT?) Vérifie si adr ln correspondent à une chaine de caractères.
(2DEBUT?) idem pour 2 chaines juxtaposées.
(LON) primitive recherche de longueur de chaîne.
LON$ laisse la longueur réelle d'une chaine.
MAXLON$ extrait la longueur maximale possible d'une chaîne.
LON$! permet de fixer la valeur de la longueur réelle d'une chaine dans la limite du maximum.
LON$+! permet l'incrément et le décrément de la longueur réelle d'une chaine.

```
\ VIDE$  +!$  !$ (PAD+) PAD$ VPAD$
: VIDE$ (S adr ln ... )
2DUP MAXLON$ >R DROP 1- R> 1+ ERASE ;
: +!$ (S adr1 ln1 adr2 ln2    ... )
   ROT >R 2DUP R@ LON$+! + R> CMOVE ;
: !$ (S adr1 ln1  adr2 ln2    ... )
  2DUP 0 LON$! DROP 0 +!$ ;
: (PAD+) ( .... )
  0 PAD 1+ C!  255 PAD 2+ C! PAD 3 + ;
: PAD$ (S ... adrpad ln ) (PAD+) COUNT ;
: VPAD$ PAD$ 0 LON$! ; ( ....... )
```

Commentaires
VIDE$ vide physiquement une chaine désignée
Utilisation de +!$
A1 ln1 A2 ln2 +!$,ajoute à la suite de la chaine A2 les caractères de A1
Utilisation de !$
A1 ln1 A2 ln2 !$ remplace les caractères de A2 par ceux de A1 (La chaîne A1 reste inchangée)
(PAD+) crée une pile de chaîne de caractères dont l'adresse de début varie avec PAD.
PAD$ laisse l'adresse et la longueur de la pile de chaînes
ATTENTION! PAD$ N'EST QU'UN TAMPON PROVISOIRE UTILISE PAR

CHAINVOC. Il faut donc récupérer rapidement son adresse sur la pile paramètres sinon on risque de perdre son contenu.
VPAD$ remet la longueur de PAD$ à zéro

```
\   >P$  (TXT) =" +" ?"
: >P$  (S adr ln ... ) PAD$ +!$ ;
: (TXT)  (S  <texte>  ... adr ln )
HERE C/L 1+  BLANK 34 WORD COUNT ;
: ="  (S  adr ln ...   )
  STATE @ IF [COMPILE] " COMPILE 2SWAP COMPILE !$
  ELSE (TXT) 2SWAP !$ THEN ; IMMEDIATE
: +"  (S  adr ln ... )
  STATE @ IF [COMPILE] " COMPILE 2SWAP COMPILE +!$
  ELSE (TXT) 2SWAP +!$ THEN ; IMMEDIATE
: ?" (S adr ln ... )  255 MIN  TYPE ;
```

Commentaires
>P$ prend une chaîne sur la pile et l'ajoute à PAD$.
(TXT) primitive de saisie de texte au clavier.
=" affecte des caractères à une chaine, soit en compilation soit en execution. Les caractères précédents sont ecrasés.
Utilisation: CHAINE =" Ceci est une chaine " (" est le caractère délimiteur)
+" ajoute du texte à une chaine donnée.
Exemple CHAINE +" de caractères "
et CHAINE ?" affiche " Ceci est une chaine de caractères "
?" affiche une chaine à l'écran.

```
\ REMPLI$   INTERC$   R"   E"
: REMPLI$ (S adr ln ascii  ---- )
  >R 2DUP MAXLON$ >R 2DUP R@ LON$! DROP R> R> FILL ;
: INTERC$ (S adr1 ln1 adr2 ln2 n --- )
  5 ?ENOUGH ABS OVER MIN >R 2DUP 4 PICK LON$+!
  R@ - SWAP R> + ROT 2DUP >R >R
  OVER + ROT CMOVE> R> R> CMOVE> ;
: R" (S adr ln n ...  )
  OVER - SPACES ?" ;
: E" (S adr ln ... )
  80 OVER - 2/ SPACES ?" ;
```

Commentaires
REMPLI$ remplis au maximum une chaine donnée avec un code ascii préalablement déposé sur la pile. Utilisation:
A1 ln1 A2 ln2 n INTERC$ permet d'inserer A1 derrière le nième caractère de A2.
R" affichage formaté à l'ecran avec alignement à droite
E" affichage equidistant des bords de l'écran

```
\ -BLG   -BLD  -BLGD  +BLG  +BLD
: -BLG (S adr lon --- )
  2DUP BL SKIP 2SWAP !$ ;
: -BLD (S adr lon --- )
  2DUP -TRAILING SWAP DROP LON$! ;
: -BLGD (S adr ln ... )
  2DUP -BLG OVER (LON) MIN -BLD ;
: +BLG (S adr lon n --- )
  >R 2DUP R@ LON$+!
PAD$ R> LON$! PAD$ BLANK 2DUP >P$ PAD$ 2SWAP !$ ;
: +BLD (S adr lon n --- )
  >R 2DUP R@ LON$+! + R> BLANK ;
```

Commentaires
-BLG: Supprime les "blancs" (code 32) au début d'une chaine et réajuste la longueur.
-BLD même effet à l'extrémité droite d'une chaine
-BLGD effet conjugué des 2 mots précédents
+BLG ajoute n espaces à gauche d'une chaine
+BLD idem pour extrémité droite

```
\  GAU$   DROI$   MIL$
: GAU$ (S adr lon n --- adr n )
  ( n positif ou negatif )
  DUP 0< IF NEGATE ROT OVER + ROT ROT - 0 MAX
  ELSE MIN THEN ;
: DROI$ (S adr lon n --- adr' n )
  ( n positif ou négatif )
 2DUP ABS MIN SWAP ?NEGATE DUP 0< IF OVER + GAU$
 ELSE >R + R@ - R> THEN ;
: MIL$ (S adr lon  n1 n2 --- adr' lon' )
  2DUP 1< SWAP 1< OR IF SWAP . . ABORT" Erreur MIL$ "
  ELSE >R OVER SWAP - 1+ DROI$ R> GAU$ THEN ;
```

Commentaires
GAU$ laisse sur la pile l'adresse et la longueur des n (ou -n) caractères à partir de la gauche .n peut être positif ou négatif.
DROI$ même effet en partant de l'extrémité droite de la chaine.
MIL$ laisse l'adresse et la longueur de n2 caractères à compter du n1eme caractère (n1 n2 positifs).

```
\  =$   <>$   >$   <$
: =$ (S adr1 lon1 adr2 lon2 --- flag t/f )
  4 ?ENOUGH ROT 2DUP =
  IF DROP COMP ELSE 2DROP 2DROP -1 THEN  0= ;
: <>$ =$ 0= ;
: >$ (S adr1 lon1 adr2 lon2 --- flag t/f )
  4 ?ENOUGH ROT 2DUP SWAP >R >R MIN COMP R> R> >
  OVER 0=  AND SWAP 1 = OR ;
: <$ (S adr1 lon 1 adr2 lon2 --- flag t/f )
  4 ?ENOUGH ROT 2DUP SWAP >R >R MIN COMP R> R> <
  OVER 0=  AND SWAP -1 = OR ;
```

Commentaires
=$ test d'égalité de 2 chaines suivant leur contenu (ascii) et leurs longueurs.
<>$ test d'inégalité.
>$ test de supériorité (ch 1 > ch 2 ?).
<$ test d'infériorité (ch 1 < ch 2 ?).

```
\  (ECH)   ECH$
: (ECH) (S adr1 lon1 adr2 lon 2 --- )
  VPAD$ 2DUP >P$ 2OVER >R >R !$ PAD$ R> R> !$ ;
: ECH$ (S  adr1 lon1 adr2 lon2 ---- )
  2DUP MAXLON$ OVER >R >R  2SWAP
  2DUP MAXLON$ OVER R> >   SWAP
  R> < OR IF ABORT" Echange impossible "
         ELSE (ECH)  THEN ;
```

Commentaires
(ECH) primitive d'interversion du contenu de 2 chaines.
ECH$ intervertit le contenu de 2 chaines de longueurs compatibles.

```
\ DANS?
: DANS?
 (S adr1 lon1 adr2 lon2 --- f si non trouve,
                         adr' lon' t si trouve )
  4 ?ENOUGH DUP 3 PICK <
    IF  2DROP 2DROP FALSE
    ELSE 2 PICK >R  OVER >R SEARCH
           IF   R> + R> TRUE
           ELSE R> R> DROP 2DROP FALSE
           THEN
    THEN ;
```

Commentaire
Utilisation: A1 ln1 A2 ln2 DANS?
Cherche la première occurence de A1 dans A2. Laisse un drapeau faux si non trouvé et ad' ln' drapeau vrai si trouvé (ad'= adresse dans A2 du début des caractères de A1).

```
\  -$       OTE$
: -$  (S adr1 lon1 adr2 lon2   ... )
    2OVER (DEBUT?) DANS?
    IF  DUP >R 2OVER + >R OVER >R + R> R>
        OVER - R@ - CMOVE
          R> NEGATE LON$+!
    ELSE 2DROP
THEN ;
: OTE$ (S adr1 lon1 adr2 lon2 --- adr/pad lon/pad )
  VPAD$ 2SWAP >P$ PAD$ 2SWAP -$ PAD$ ;
```

Commentaires
-$ supprime d'une chaine A1 la première occurence qui soit égale à la chaine A2. Ne fait rien si la chaine n'est pas trouvée. Exemple:

```
  CH1=" ESSAI DE CHAINE "
       CH2 =" ESSAI DE "
  CH1 CH2 -$    CH1 devient " CHAINE "
```

La longueur de A1 est réajustée.
OTE$ met le résultat de CH1 - CH2 dans PAD$ et laisse cette adresse sur la pile.Les 2 chaines ne sont pas modifiées.

```
\ Mots de concatenation (( et ))
: ((  ( ---- )
  ?COMP COMPILE VPAD$ COMPILE 0 COMPILE 0 9 ;
  IMMEDIATE
: ))  (S  adr1/ln1 à adrx/lnx   --- adrpad lonpad )
  9 ?PAIRS
  [COMPILE] BEGIN COMPILE OVER [COMPILE] IF
  COMPILE PAD$ COMPILE 0 COMPILE INTERC$ [COMPILE] THEN
  COMPILE OVER COMPILE 0= [COMPILE] UNTIL
  COMPILE 2DROP COMPILE PAD$ ; IMMEDIATE
```

Commentaires
((début de concaténation de chaines dans PAD$.
)) met fin à la concaténation de chaines et laisse sur la pile l'adresse et la longueur du résultat (PAD$). Exemple: si A$ B$ C$ D$ sont des chaines comptées, la séquence ((A$ B$ C$ D$)) X$!$ concatène les 4 chaines et met le résultat dans la chaine X$, dans cet ordre.
Utilisables en compilation uniquement

```
\ Conversions 2VAL  VAL D>CH$  S>CH$  C@$  KEY$
: 2VAL (S adr ln ... d )
  VPAD$ >P$ BL PAD$ + C! (PAD+) NUMBER? DROP ;
: VAL (S adr ln ... n ) 2VAL DROP ;
: D>CH$ (S d --- adr lon )
  (D.) VPAD$ >P$ PAD$ ;
: S>CH$ (S n ... adr ln ) S>D D>CH$ ;
: C@$   (S adr ln ... ascii ) DROP C@ ;
: KEY$  (S touche ... adr ln ) KEY BL MAX S>CH$ ;
```

Commentaires
2VAL convertit une chaine en un nombre double longueur. Laisse 0 0 si impossibilité.
VAL idem mais laisse un nombre en simple longueur.
D>CH$ prend un nombre en DL sur la pile et le transforme en chaine de caractères envoyée dans PAD$. Laisse adpad et lnpad.
S>CH$ idem avec un nombre en SL.
C@$ laisse le code ascii du 1er caractère d'une chaine explicite.
KEY$ attend la frappe au clavier d'un caractère Ascii et le transforme en une chaine de 1 caractère.

```
\ Fonctions INPUT$   INKEY$
: INPUT$ (S n --- adrpad lonpad )
  1 ?ENOUGH VPAD$ 255 MIN
  PAD$ DROP DUP >R SWAP ." ? "
  EXPECT SPACE SPAN @
  R> 1- C! PAD$ ;
: INKEY$ (S adr lon --- )
  2 ?ENOUGH OVER SWAP (KEY?) IF 1 LON$! 0 8 BDOS SWAP C!
  ELSE 0 LON$! 0 SWAP C! THEN ;
```

Commentaires
INPUT$ attend la frappe au clavier d'au plus n caractères. Laisse le résultat sur la pile sous forme de chaine de caractères (PAD$). Exemple:
  20 INPUT$ A$!$ affecte les caractères frappés au clavier à la chaine A$.
INKEY$ permet de saisir "au vol" au clavier un caractère ascii et l'affecte à une chaine. Exemple: A$ INKEY$

```
\ MET$   SPACE$   SEPAR$   MOT$   ?MOT$
: MET$ (S adr lon n ascii --- )
  4 ?ENOUGH >R >R 2DUP MAXLON$ R> MIN >R
  2DUP R@ LON$!  DROP R> R> FILL ;
: SPACE$ (S adr lon n ... ) BL MET$ ;
 VARIABLE SEPAR$      BL SEPAR$  !
: MOT$ (S adr lon n --- adr' lon' )
  3 ?ENOUGH    1 MAX >R TUCK R> 0 DO
  ROT DROP SEPAR$ @ SKIP OVER SWAP SEPAR$ @ SCAN
  LOOP DROP OVER - ;
: ?MOT$ (S adr lon n --- adr' lon' flag v
                   ou bien flag f )
  MOT$ DUP 0= IF 2DROP FALSE ELSE TRUE THEN ;
```

Commentaires
MET$ met n fois le code ascii dans une chaine.
SPACE$ met n espaces (code 32) dans une chaine.
SEPAR$ variable contenant un code ascii servant au mot suivant MOT$. A l'initialisation contient le code 32.
MOT$ extrait le nieme mot d'une chaine, limité par le code
ascii contenu dans SEPAR$. Exemple: Soit A$ =" A B C D E "
A$ 3 MOT$ laisse sur la pile l'adresse et la longueur du caractère "C".
?MOT$ fait la même chose, mais avec un test de présence. Laisse adr ln drapeau vrai si trouvé, sinon drapeau faux.

```
\  Interprétation et exécution d'une chaine (JEDI)
0 0 2CONSTANT STR$  VARIABLE (>IN)
: ?ERROR$ ( adr lon flag  - - - )
DUP  IF ['] (SOURCE) IS SOURCE
        ['] (?ERROR) IS ?ERROR
        (>IN) @ >IN !
THEN (?ERROR) ;
: EXECUTE$ ( adr lon  --- )
['] STR$ >BODY 2!
['] STR$       IS SOURCE
['] ?ERROR$     IS ?ERROR
>IN @ (>IN) ! >IN OFF  RUN
['] (?ERROR) IS ?ERROR
['] (SOURCE) IS SOURCE
(>IN) @ >IN ! ;

\ PLACE$
: PLACE$ (S adr1 lon1 adr2 lon2 n  ... )
  1- 0 MAX >R (DEBUT?) 2 PICK R@ +
  2DUP < IF 2 PICK >R
            LON$! R>
         ELSE  2DROP
  THEN
  R> + SWAP MOVE ;
```

Commentaire
PLACE$ inscrit le contenu de la chaine A1 dans la chaine A2, à partir du nieme caractère de A2.Les caractères précédents sont écrasés. Exemple:
   A1$ A2$ 10 PLACE$

```
\ OCCUR$
: OCCUR$ (S  Adr lon --- n )
  DUP        IF
             0 >R TUCK
      BEGIN
        ROT DROP SEPAR$ @
        SKIP
        OVER SWAP SEPAR$ @
        SCAN
        DUP 0>   WHILE R> 1+ >R
      REPEAT
             DROP 2DROP R>
             ELSE   2DROP
  0 THEN   ;
```

\ Commentaire
Laisse sur la pile le nombre de mots (d'occurences...) délimités par le code ascii contenu dans la variable SEPAR$. Le résultat est 0 si la chaine est vide ou bien ne contient aucun séparateur ou encore si la chaine n'est constituée que du code du séparateur.

```
\ Structures de contrôle CASE$ OF$ ENDOF$ ENDCASE$
: CASE$ (S adr ln ... )
  ?COMP CSP @ !CSP 2 ; IMMEDIATE
: NOCASE$ 2DUP ; (S adr ln ... adr ln adr ln )
: OF$ (S adr1 ln1 adr2 ln2 ... adr1 ln1 )
  2 ?PAIRS     COMPILE 2OVER  COMPILE =$  COMPILE ?BRANCH
  >MARK        COMPILE 2DROP 3 ; IMMEDIATE
: ENDOF$
  3 ?PAIRS      COMPILE BRANCH  >MARK
```

```
  SWAP >RESOLVE 2 ; IMMEDIATE
: ENDCASE$ (S adr ln .... )
  2 ?PAIRS COMPILE 2DROP  BEGIN  SP@ CSP @ <> WHILE
  >RESOLVE REPEAT CSP ! ; IMMEDIATE
```

Commentaire
Structures de contrôle appliquées aux chaînes de caractères. Exemple d'utilisation:

```
  : TEST   CH1
  CASE$
  CH2  OF$    FAIRE A ENDOF$
  CH3  OF$     FAIRE B ENDOF$
  ENDCASE$   ;
  Execute FAIRE A si CH1=CH2
  Execute FAIRE B si CH1=CH3
```

```
\  ROD$  MAJUSC
: ROD$ (S adr lon ... )
1- 2DUP + C@ >R >R DUP DUP 1+ R> MOVE R> SWAP C! ;
: MAJUSC (S adr lon ... )
(DEBUT?) BOUNDS    DO  I C@
       CASE  ASCII é =OF ASCII E ENDOF
             ASCII è =OF ASCII E ENDOF
             ASCII à =OF ASCII A ENDOF
             ASCII ç =OF ASCII C ENDOF
             ASCII ù =OF ASCII U ENDOF
        NOCASE    ENDCASE
UPC I C!           LOOP ;
```

Commentaire
ROD$ effectue une rotation circulaire sur une chaine par la droite (Met le dernier caractère devant le premier). A titre de curiosité essayez ce petit programme de démonstration:

```
  40 VCHAIN A$
  A$ =" J E   M E  D E F I L E "
  : DEFILE
  DARK 50 0 DO A$ 2DUP 25 2 AT  ROD$ 200 MS LOOP ;
```

MAJUSC transforme les minuscules en majuscules, accentués compris.

```
\ De majuscule en MINUSC   ( sans accentuation !)
: MINUSC (S  adr lon ... )
  (DEBUT?)  BOUNDS
  DO   I C@ DUP 65 90 BETWEEN
    IF BL OR I C! ELSE DROP  THEN
  LOOP  ;
```

Commentaire
MINUSC transforme les caractères majuscules en minuscules (sans accentuation bien sûr...)

```
\  -BLSUP    -SCAN
: -BLSUP (S adr lon ... )
(DEBUT?) 2DUP -BLG OVER (LON) MIN 2DUP >R >R VPAD$
BEGIN 2DUP     BL SCAN
        2SWAP 2 PICK - 1+  >P$
        BL SKIP   DUP 0=
UNTIL 2DROP  PAD$ 1-   -TRAILING  R> R> !$ ;

: -SCAN (S  adr lon ascii ...adr' lon' )
  ROT ROT DUP 0 DO
    2DUP + 1- C@ 3 PICK <>
    IF  1-
    ELSE   LEAVE    THEN   LOOP
   ROT DROP  ;
```

Commentaires
-BLSUP supprime d'une chaîne tous les espaces superflus ou jugés comme tels.
-SCAN réalise un SCAN en commençant par la fin (comme un -TRAILING passe-partout)

```
\ Affiche centré d'une chaine par C"
VARIABLE CAR/LIGNE     40 CAR/LIGNE !
: C"  (S  adr lon .... )
(DEBUT?)  2DUP  -BLSUP DROP DUP (LON)
BEGIN  DUP CAR/LIGNE C@  >
WHILE
           >R DUP CAR/LIGNE C@ + C@ BL = NOT
           IF     CAR/LIGNE  C@ BL -SCAN DUP 0=
                     IF DROP CAR/LIGNE C@  THEN
           ELSE   CAR/LIGNE C@           THEN
           2DUP R> SWAP - >R  OVER + >R
           E" CR  R> R>
REPEAT     E"    ;
```

Commentaire
CAR/LIGNE variable servant au mot C". Initialisé à 40, C" affiche une chaine en restant equidistant des bords de l'écran, sur une largeur maximale contenue dans CAR/LIGNE.

```
\ Ne sont pas incluses au sein de ces écrans les
\ définitions de ?COMP et ?PAIRS     ( Voir FIG-FORTH )
\ Les voici:
: ?COMP
  STATE @ 0= IF ABORT" Sytème non en compilation " THEN ;
: ?PAIRS
  - 0= NOT IF ABORT" Mauvais emboîtement de structure"
  THEN ;
```

Ndlr: Nous avons été obligé, pour des raisons de place de modifier la présentation initiale de ce programme afin de le faire tenir dans les quelques pages encore disponibles de ce présent numéro.

REPONSES: concernant la gestion de la totalité de la mémoire des systèmes IBM PC et compatibles, TURBO-Forth utilise des primitives dont voici résumé brièvement les caractéristiques:

```
 LC!       c seg off ---
 LC@       seg off --- c
 L!        n seg off ---
 L@        seg off --- n
 LCMOVE    seg1 off1 seg2 off2 long ---
 LCMOVE>   seg1 off1 seg2 off2 long ---
```

Nos remerciements à Mr JACCOMARD pour les définitions de ces mots créées pour l'amélioration de FORTHLOG II et intégrées à TURBO-Forth. Les mots LCMOVE et LCMOVE> initialement écrits en FORTH ont depuis été réécrits en assembleur 8086.

Attention, il peut être risqué de modifier une zone mémoire utilisée par un autre programme. Exemple, sous TURBO-Forth, si vous tapez PROGRAM TURBO.COM, vous serez à nouveau sous le contrôle de TURBO-Forth, mais dans une copie située à un autre endroit de la mémoire. Pour plus de clarté, appelons T1 le premier TURBO lancé au démarrage du système; vous lancez T2 à partir de T1; T1 reste en mémoire mais vous ne savez pas où!. L'abandon de T2 par BYE vous renvoie à T1. Avec 640k de mémoire, jusqu'à six exemplaires de TURBO peuvent cohabiter en mémoire. Sachant que TURBO peut exécuter un ordre depuis MSDOS comme suit:

```
  TURBO WORDS
```

ou depuis TURBO

```
  " WORDS" PASS PROGRAM TURBO.COM
```

et en version compilée:

```
: mot ....
 ... " WORDS" PASS
  " PROGRAM TURBO.COM" $EXECUTE
 .... ;
```

je vous laisse imaginer les fantastiques possibilités qui s'offrent à vous en gérant astucieusement des bibliothèques de routines compilées en FORTH. Je présenterai dans le prochain numéro la première routine INTER-LOGICIELLE programmée en FORTH:

- menus déroulants pour programmes .BAT.
- menus déroulants pour programmes dBASE III/III+.